UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

VICTOR DOS SANTOS MELO

Sistema Web para Compartilhamento de Conteúdo sobre Ciência da Computação

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Marcelo Soares Pimenta

Porto Alegre
2021

Dedico este trabalho à minha mãe e
ao meu pai (*in memorian*), por serem
meus melhores exemplos de superação
e da importância da educação.
Também dedico à minha esposa Débora,
principalmente pela paciência nos
momentos em que estive ausente para
atuar nesse projeto.

AGRADECIMENTOS

Toda pessoa tem seus méritos ao conquistar objetivos. Porém, nunca pode dizer que conquistou tudo sozinho. Toda jornada é apoiada direta ou indiretamente por múltiplas outras pessoas, e aqui quero deixar meus agradecimentos àquelas que apoiaram a minha.

Primeiramente agradeço aos meus pais, meu irmão e minha esposa. Vocês me apoiaram de tantas formas que é impossível enumerá-las, e por isso serei eternamente grato.

Agradeço ao professor Marcelo Soares Pimenta pela excelente orientação nesse trabalho e pela oportunidade de ter começado ele já no início do ano, durante uma disciplina. Graças a isso consegui desenvolver um sistema que não só vai ser útil para meus propósitos após a universidade, mas que também me permitiu aprender diversas tecnologias importantes para minha carreira como engenheiro de software.

Agradeço aos professores que tive no ensino superior. A forma com que vocês apresentaram o conteúdo e responderam minhas dúvidas após as aulas foi a razão principal de eu ter vindo para este curso e ter me mantido motivado nele durante todo período aqui.

Também agradeço aos colegas que tive, tanto aqueles que estiveram ao meu lado quando eu estava em outra universidade quanto os que estiveram comigo na UFRGS. As discussões extraclasse sobre as matérias foram momentos muito importantes para o aprofundamento do conteúdo.

Agradeço à todos que me deram oportunidade para me desenvolver fora da sala de aula durante esse período: ao professor Rafael H. Bordini que, no começo da minha jornada em ciência da computação, me introduziu à pesquisa acadêmica através da iniciação científica; aos professores e responsáveis pelo programa do Apple Developer Academy - PUCRS, onde pude adquirir conhecimentos que me abriram as portas para o mercado de trabalho como desenvolvedor iOS; e por fim aos meus colegas de trabalho que adoram debater sobre engenharia de software, me mostrando que mesmo após a universidade ainda haverá muito espaço para essas conversas.

RESUMO

Este documento apresenta o desenvolvimento de um sistema web no formato de blog para o compartilhamento de conteúdo sobre ciência da computação. Esse sistema contém dois fluxos principais: um para o usuário leitor e outro para o usuário administrador. Além das publicações tradicionais de blogs, o sistema consta com tagueamento por assunto, tempo estimado de leitura, área de comentários, uma barra de busca, uma página "sobre mim", entre outras funcionalidades menores. Esse sistema é hospedado utilizando uma máquina virtual em nuvem que inicialmente continha apenas o sistema operacional, sendo necessário um trabalho para converte-la em um servidor web totalmente acessível via HTTPS. Para isso foram usadas tecnologias atuais como `Node.js`, AWS, React, entre outras.

Palavras-chave: Blog. engenharia de software. node. react. AWS. NoSQL.

Web system for sharing computer science content

ABSTRACT

This document presents the development of a web system in blog format for sharing computer science content. This system contains two main streams: one for the reader user and one for the admin user. In addition to traditional blog posts, the system includes tagging by subject, estimated reading time, comments area, a search bar, an "about me" page, among other minor features. This system is hosted using a cloud virtual machine that initially contained only the operating system, requiring work to convert it into a fully accessible web server via HTTPS. For this, current technologies such as `Node.js`, AWS, React, among others, were used.

Keywords: Blog, software engineering, node, react, AWS, NoSQL.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AWS	Amazon Web Services
CORS	Cross-Origin Resource Sharing
CSS	Cascading Style Sheets
EC2	Elastic Compute Cloud
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JWT	JSON Web Token
OGP	Open Graph Protocol
REST	Representation State Transfer
S3	Simple Storage Service
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UX	User Experience

SUMÁRIO

1 INTRODUÇÃO

Ensinando, aprende-se. Esta frase foi atribuída à Sêneca, filósofo estóico romano. Uma das coisas mais importantes que podemos fazer quando estamos tentando aprender assuntos como matemática ou ciência é trazer ideias abstratas à vida em nossas mentes (OAKLEY, 2014). Em outras palavras, é materializar um conteúdo abstrato de forma que seja mais fácil entendê-lo.

Scott Young, autor de *Ultralearning* (YOUNG; CLEAR, 2019), apresenta a **Técnica Feynman** como um modelo de aprendizado baseado no ensino (LEARN..., 2011). Ao tentar entender um conteúdo usando essa técnica, deve-se escrever sobre ele como se estivesse-o explicando para alguém totalmente leigo no tema. O cérebro precisa compreender um conteúdo para simplificá-lo a ponto de conseguir explicá-lo a um leigo, e é assim que essa técnica nos ajuda a parar de reproduzir o que foi lido e começar a realmente entender sobre o tema.

A partir dessa técnica e de aplicações empíricas, a maior motivação para a criação deste blog foi poder compartilhar sobre conteúdos que estou estudando, e por consequência também auxiliar outras pessoas da área a ampliarem seus conhecimentos.

Vale aqui também comentar que, em uma época onde as redes sociais estão altamente integradas em nosso dia-a-dia, é comum usuários compartilharem conteúdos excelentes dentro delas. Porém, ainda que desconsiderássemos as motivações já citadas, o formato em blog traz um aspecto mais profissional, fortalecendo visualmente a conexão entre o autor e o conteúdo. Vantagens como essa fazem com que, mesmo com as redes sociais, o formato em blog siga sendo amplamente utilizado, dado os inúmeros blogs existentes sobre os mais diversos assuntos.

1.1 Objetivos

O objetivo desse trabalho é mostrar como foi a implementação de um sistema web estilo blog. Esse sistema foi planejado para contemplar dois fluxos distintos: o fluxo do usuário leitor, que acessa o sistema para interagir com o conteúdo disponível; e o fluxo administrador, que acessa o site para realiza o gerenciamento do conteúdo. Além desses fluxos, o sistema também possui uma página "sobre mim", com informações e links para as redes sociais do administrador.

Esse trabalho também descreve como foi feita a configuração da infraestrutura

utilizando serviços em nuvem, incluindo uma máquina virtual sobre a qual o sistema é executado.

1.2 Estrutura do trabalho

Esse trabalho está estruturado em 7 capítulos: no Capítulo 1 é apresentada a motivação dele e traz seus objetivos gerais; o Capítulo 2 lista uma série de fundamentos necessários para o entendimento completo dele; no Capítulo 3 são listados alguns trabalhos relacionados que inspiraram este; o Capítulo 4 exibe a metodologia utilizada no desenvolvimento do trabalho; o Capítulo 5 apresenta o resultado final obtido, descrevendo-o usando uma abordagem *top-down*; no Capítulo 6 é apresentado um teste de usabilidade respondido por 146 pessoas e as respostas obtidas; finalmente o Capítulo 7 resume o que foi feito, suas limitações e trabalhos futuros.

2 CONCEITOS E FUNDAMENTOS

Este capítulo apresenta os conceitos necessários para o entendimento do resto deste trabalho. Ele pode ser lido na ordem ou ser usado como consulta quando referenciado nos próximos capítulos. Os conceitos estão ordenados em ordem alfabética para facilitar a consulta.

2.1 Amazon Web Services (AWS)

AWS é um conjunto de plataformas da Amazon que provê diversos *web services on-demand* em nuvem para pessoas físicas, companhias e governos. A estratégia de pagamento é do tipo *pay-as-you-go*. Estes *web services* são abstraídos de tal forma que podem ser conectados como blocos, onde um serviço ao finalizar chama outros, passando ou não seu valor de saída. A tecnologia da AWS é implementada em *server farms* ao redor do mundo. Mais informações podem ser vistas em (AMAZON, 2021f).

2.2 Amazon Elastic Compute Cloud (EC2)

Amazon Elastic Compute Cloud (EC2) é um serviço que provê instâncias de máquinas virtuais na nuvem. Essas máquinas podem ser acessadas via linha de comando, e seu gerenciamento é feito através do *dashboard* no site da AWS. Existem diversos tipos de instâncias que podem ser contratadas, e como exemplo podemos citar as instâncias do tipo T2 que são de uso geral e possuem baixo custo. A tabela na Figura 2.1 exibe os valores mostrados no site da AWS (AMAZON, 2021e) para esse tipo de instância. Mais informações podem ser vistas em (AWS, 2021a).

Figura 2.1: Detalhes das diferentes instâncias T2.

Nome	vCPUs	RAM (GiB)	Créditos de CPU/h	Preço sob demanda/hora*	Instância reservada por 1 ano – por hora*	Instância reservada por 3 anos – por hora*
t2.nano	1	0,5	3	0,0058 USD	0,003 USD	0,002 USD
t2.micro	1	1,0	6	0,0116 USD	0,007 USD	0,005 USD
t2.small	1	2.0	12	0,023 USD	0,014 USD	0,009 USD
t2.medium	2	4,0	24	0,0464 USD	0,031 USD	0,021 USD
t2.large	2	8,0	36	0,0928 USD	0,055 USD	0,037 USD
t2.xlarge	4	16,0	54	0,1856 USD	0,110 USD	0,074 USD
t2.2xlarge	8	32,0	81	0,3712 USD	0,219 USD	0,148 USD

2.3 Amazon Simple Email Service (SES)

Amazon Simple Email Service (SES) é um serviço de e-mail econômico, flexível e escalável que permite aos desenvolvedores enviar mensagens de qualquer aplicativo. É possível configurar o SES rapidamente para oferecer suporte a vários casos de uso de e-mail, incluindo comunicações transacionais, de marketing ou para disparar e-mail em massa.

2.4 Amazon Simple Storage Service (S3)

Amazon Simple Storage Service (S3) é um serviço de armazenamento de objetos que oferece escalabilidade, disponibilidade, segurança e desempenho líderes do setor. Isso significa que clientes de todos os tamanhos e setores podem usá-lo para armazenar e proteger qualquer quantidade de dados para uma variedade de casos de uso, como *data lakes*, sites, aplicativos móveis, backup e restauração, arquivamento, aplicativos corporativos, dispositivos IoT e realizar grandes análises de dados. O S3 oferece recursos de gerenciamento fáceis de usar para que seja possível organizar os dados e configurar controles de acesso ajustados para atender à diversos requisitos específicos de negócios, organizacionais e de conformidade. O S3 foi projetado para 99,999999999% (11 9's) de durabilidade e armazena dados para milhões de aplicativos para empresas em todo o mundo. Mais informações podem ser vistas em (AWS, 2021b).

2.5 AWS-sdk

AWS-sdk é o SDK utilizado para comunicação através do código com recursos da AWS. Ele é fornecido para diferentes linguagens, como C++, Go, Java, JavaScript, Python, entre outras. Mais informações podem ser vistas em (AWS, 2021c).

2.6 Axios

Axios é um cliente HTTP baseado em *promises* para `Node.js` e navegador. Ele é isomórfico, significando que pode ser executado tanto no navegador quanto no servidor `Node.js` com a mesma base de código. No lado do servidor, ele usa o módulo http na-

tivo do `Node.js`, enquanto no navegador ele usa XMLHttpRequests. Mais informações podem ser vistas em (AXIOS, 2021; AXIOS, 2021).

2.7 Banco de Dados NoSQL

Bancos de dados NoSQL são bases de dados não-relacionais com esquemas flexíveis. Eles são amplamente reconhecidos por sua facilidade de desenvolvimento, funcionalidade e performance em escala. Ao contrário do que pode ser assumido inicialmente, bancos NoSQL podem ser usados para lidar com objetos que possuem relacionamentos. Porém, esses relacionamentos devem ser gerenciados pela aplicação, diferentemente de bancos relacionais como MySQL ou PostgreSQL que gerenciam esses relacionamentos sozinhos.

Bancos NoSQL estão na forma não normalizada (*unnormalized form*). Sendo projetados assim, eles dão prioridade para escalabilidade e mutabilidade, sacrificando aquelas características ganhas com as formas normais (e.g., 1NF, 4NF, 6NF). Mais informações podem ser vistas em (MONGODB, 2021c).

2.8 Blog

Um **blog** (truncamento de *weblog*) é um site informativo ou de discussão publicado na internet consistindo de publicações discretas em texto. Estas publicações são tipicamente apresentadas em ordem cronológica inversa, de tal forma que a última publicação fique no topo da página (WIKIPEDIA, 2021a). Em 2018 já existiam mais de 440 milhões de blogs, contando apenas aqueles publicados pelo Tumblr, Squarespace e Wordpress (UNIVERSITY, 2018).

2.9 Bootstrap

Bootstrap é um *framework* que permite projetar e personalizar rapidamente sites responsivos para dispositivos móveis. É um kit de ferramentas de front-end de código aberto, apresentando variáveis e mixins Sass, sistema de grade responsivo, extensos componentes pré-construídos e poderosos plug-ins de JavaScript. Mais informações podem ser vistas em (BOOTSTRAP, 2021a).

2.10 Certbot

Certbot é uma ferramenta gratuita e *open source*, cujo objetivo é gerenciar automaticamente certificados do **Let's Encrypt**, permitindo o uso de HTTPS em sites que são administrados manualmente, sem auxílio de ferramentas providas por empresas de hospedagem. Mais informações podem ser vistas em (FOUNDATION, 2021a; (ISRG), 2021).

2.11 DevOps

DevOps é a combinação de filosofias, práticas e ferramentas que aumentam a habilidade de uma organização em entregar aplicações e serviços com alta velocidade. Com DevOps as organizações conseguem evoluir e melhorar produtos em um ritmo mais rápido do que se utilizassem processos tradicionais de gerenciamento de infraestrutura e software. Esta velocidade permite a elas servirem melhor seus clientes e competirem mais efetivamente no mercado. No modelo DevOps, os times de desenvolvimento e operações não são mais isolados entre si, permitindo que processos possam ser definidos e desenvolvidos em conjunto. Mais informações podem ser vistas em (AWS, 2021d).

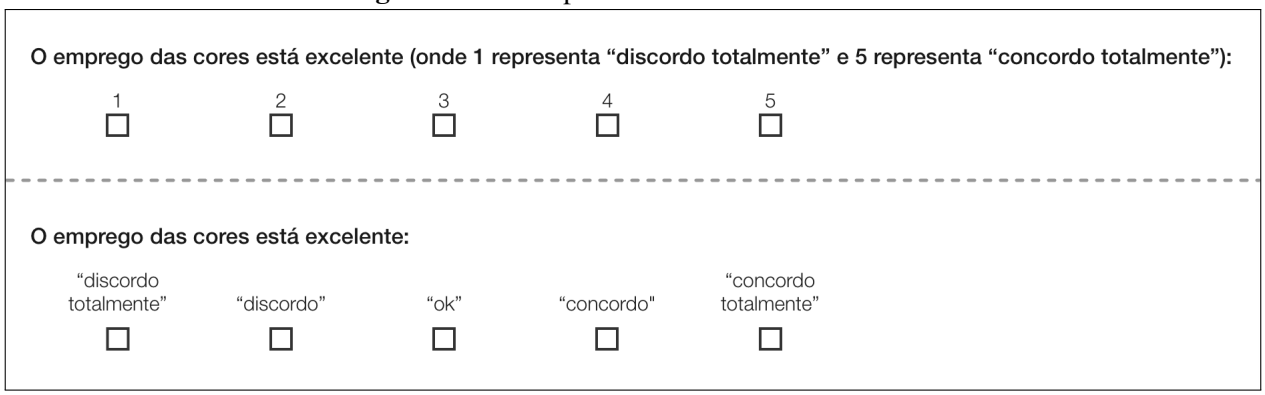
2.12 Dotenv

Dotenv é um módulo que carrega variáveis de um arquivo `.env` na propriedade `process.env` do `Node.js`. Armazenar as configurações de ambiente separadas do código é uma abordagem sugerida pela metodologia *The Twelve-Factor* (WIGGINS, 2017). Mais informações podem ser vistas em (MOTDOTLA, 2021).

2.13 Escala de Likert

Segundo (ROGERS; SHARP; PREECE, 2013), as **escalas de Likert** são utilizadas para medir opiniões, atitudes e crenças e, consequentemente, são amplamente utilizadas para avaliar a satisfação dos usuários com relação a produtos. Essa avaliação é feita utilizando diferentes afirmações e questionando o usuário sobre o quanto ele concorda com tais afirmações. Por exemplo, as opiniões dos usuários sobre o uso de cores em um

Figura 2.2: Exemplo de uso da escala de Likert.



site podem ser avaliadas com uma escala de Likert utilizando uma faixa de números ou palavras, conforme mostra a Figura 2.2.

2.14 Express

Express é o *framework* para `Node.js` mais popular, sendo usado inclusive por inúmeros outros *frameworks*. Ele provê mecanismos para:

- Definir *handlers* de requisições para cada endpoint HTTP.
- Integração com *engines* de renderização de *view*, fornecendo os dados a serem exibidos.
- Definir configurações comuns de aplicações web, como a porta usada.
- Adicionar *middlewares* no processamento das requisições, em qualquer etapa do tratamento da requisição.

Mais informações sobre Express podem ser vistas em (MOZILLA, 2021a; COMMUNITY, 2021a).

2.15 Express-cors

Express-cors é um pacote `Node.js` com objetivo de prover um *middleware* para o Express que irá agir como um CORS (Cross-Origin Resource Sharing). Com essa biblioteca é possível gerenciar o controle de acesso aos recursos do servidor. Mais informações podem ser vistas em (JS, 2021a).

2.16 Express-jwt

Express-jwt provê um *middleware* para Express com objetivo de validar e decodificar JWTs (Seção 2.22) enviados pelo cliente. Mais informações podem ser vistas em (AUTH0, 2021).

2.17 Express-validator

Express-validator é um *middleware* para Express que sanitiza e valida dados de entrada. Sanitizar um dado é garantir que ele está livre de ruídos e no formato adequado, garantindo que só vai executar as próximas validações em dados sintaticamente corretos. Mais informações podem ser vistas em (VALIDATOR, 2021).

2.18 Formidable

Formidable é um módulo para `Node.js` para fazer *parsing* de dados recebidos através de um formulário, principalmente *uploads* de arquivos. Mais informações podem ser vistas em (FORMIDABLE, 2021).

2.19 Git e Github

Git é um controlador de versões grátis e open-source, projetado para lidar com projetos de todos os tamanhos com eficiência. Já o **Github** é um serviço de hospedagem para projetos em desenvolvimento que utiliza o Git. O Github oferece todas funcionalidades do Git, além de algumas próprias. Mais informações podem ser vistas em (CONSERVANCY, 2021; GITHUB, 2021).

2.20 HTTP

Hypertext Transfer Protocol (HTTP) é um protocolo da camada de aplicação para transferir documentos de hipermídia, como HTML. Ele foi projetado para comunicação entre navegadores e servidores web, mas também pode ser utilizado para outros propósi-

tos. HTTP segue o modelo clássico cliente-servidor, com o cliente fazendo as requisições e o servidor enviando as respostas. HTTP é um protocolo *stateless*, significando que o servidor não mantém nenhum dado (estado) entre duas requisições. Mais informações podem ser vistas em (MOZILLA, 2021b).

2.21 JS-cookie

JS-cookie é uma API JavaScript simples e leve para lidar com cookies no cliente. Ela possui diversas características importantes, incluindo:

- Funciona em todos os navegadores;
- Aceita qualquer caractere;
- Sem dependências com bibliotecas terceiras;
- Complacente com a RFC 6265;
- Permite codificação e decodificação customizada;
- Ocupa menos de 800 bytes quando gzipada.

Mais informações sobre JS-cookie podem ser vistas em (JS-COOKIE, 2021).

2.22 JSON Web Token (JWT)

JWT é um padrão de segurança da indústria definido na RFC 7519 (AUTH0, ; JONES; BRADLEY; SAKIMURA, 2015). A informação contida em um JWT pode ser verificada e confiada por ser assinada digitalmente usando uma criptografia assimétrica. JWT pode ser usado em diferentes cenários, sendo os mais comuns para autorização e troca de dados. Mais informações podem ser vistas em (AUTH0, 2021).

2.23 Marked

Marked é um compilador de markdown para HTML. Ele é de baixo nível, construído com foco em velocidade e sem cache ou bloqueio por longos períodos. Ele funciona no navegador, servidor ou até via linha de comando. Mais informações podem ser vistas em (MARKED, 2021a; MARKED, 2021b).

2.24 Markdown

Markdown é uma linguagem de marcação leve criada em 2004 por John Gruber e Aaron Swartz. Seu apelo se dá pela facilidade de leitura, e é amplamente utilizada para *blogging*, mensagens instantâneas, fórums online, softwares colaborativos, páginas de documentação e arquivos `README` (GRUBER, 2004; WIKIPEDIA, 2021b).

2.25 MongoDB

MongoDB é um banco de dados NoSQL no formato de documentos: um banco é chamado de coleção, e cada coleção é composta por documentos que são os registros do banco. MongoDB foi construído visando velocidade, escalabilidade e disponibilidade. Mais informações podem ser vistas em (MONGODB, 2021b).

2.26 MongoDB Atlas

MongoDB Atlas é o serviço que permite a criação e gerenciamento de bancos de dados do MongoDB na nuvem. Atlas é disponibilizado em mais de 70 regiões através da AWS, GCP e Azure. Possui diferentes ferramentas de automação e práticas a fim de garantir disponibilidade, escalabilidade e estar em conformidade com os maiores requisitos de segurança e privacidade. Mais informações podem ser vistas em (MONGODB, 2021a).

2.27 Mongoose

Mongoose é um módulo `Node.js` para modelar esquemas do MongoDB, e ele inclui diversas funcionalidades importantes como validações, *queries*, *hooks* de lógica de negócio, entre outras. Mais informações podem ser vistas em (JS, 2021b).

2.28 Mongoose-paginate-v2

Mongoose-paginate-v2 é uma biblioteca de paginação customizada para o Mongoose (Seção 2.27). O uso principal dessa biblioteca é para paginar os dados diretamente na *query*, sem precisar de qualquer código extra na aplicação para realizar essa paginação. Mais informações podem ser vistas em (ARAVINDNC, 2021).

2.29 Morgan

Morgan um módulo `Node.js` que funciona como um *middleware* para fazer log das requisições HTTP. Com ele, podemos visualizar informações relevantes sempre que um cliente envia uma requisição ao servidor. Mais informações podem ser vistas em (EXPRESSJS, 2021).

2.30 Next.js

Ao utilizar React (Seção 2.37), existem diversos detalhes a serem considerados:

- O código precisa ser empacotado usando um empacotador como `webpack` e transformado usando um compilador como Babel;
- É necessário fazer otimizações de produção, como separação de código;
- Você pode querer pré-renderizar algumas páginas por performance e SEO. Você também talvez queira usar renderização do lado do servidor ou do lado do cliente;
- Pode ser necessário escrever algum código do lado do servidor para conectar o aplicativo React ao banco de dados.

Next.js é um *framework* que provê uma solução para todos os problemas acima. Ele contém muitas funcionalidades nativas como: roteamento baseado em página, pré-renderização, renderização do lado do servidor, recarregamento rápido, entre outras funcionalidades. Mais informações podem ser vistas em (JS, 2021c).

2.31 Nginx

NGINX é um software *open source* que pode funcionar como *web server*, proxy reverso, cache, balanceador de carga, streaming de mídia, entre outros. Inicialmente foi projetado apenas para servir como um *web server* de máxima performance e estabilidade. Mais informações podem ser vistas em (NETWORKS, 2021).

2.32 Node.js

Node.js é um ambiente de *runtime open-source* e *cross-platform* para JavaScript. Ele é uma ferramenta popular para praticamente qualquer tipo de projeto, e roda fora de um navegador através da *engine* JavaScript V8. Essa *engine* permite ao `Node.js` ser muito performático, além de permitir o uso do JavaScript para implementação do **web server**. Isso segue o paradigma de *"JavaScript everywhere"*. Mais informações podem ser vistas em (FOUNDATION, 2021c; FOUNDATION, 2021b).

2.33 Nodemon

Nodemon é uma ferramenta que auxilia o desenvolvimento com `Node.js` ao reiniciar automaticamente a aplicação sempre que detectar alterações no código. Mais informações podem ser vistas em (REMY, 2021).

2.34 Nprogress

Nprogress é uma barra de progresso minimalista inspirada no Google, Youtube e Medium. Mais informações podem ser vistas em (NPROGRESS, 2021).

2.35 Open Graph Protocol (OGP)

Open Graph Protocol (OGP) é um padrão para permitir que qualquer página se torne um objeto rico em um grafo social. Como exemplo, o Facebook utiliza este padrão para permitir que páginas web referenciadas dentro da plataforma sejam exibidas como

qualquer outro objeto interno, com título, imagem, descrição, entre outros atributos. Mais informações podem ser vistas em (FACEBOOK, 2021).

2.36 Prism.js

Prism é uma ferramenta de *highlight* de sintaxe leve e extensível, desenvolvido com os padrões de web modernos em mente. Ele é usado em milhões de sites, incluindo o site da Smashing Magazine, do MySQL e Drupal. Prism executa em quase todos os navegadores e versões de `Node.js`. Mais informações podem ser vistas em (PRISMJS, 2021a; PRISMJS, 2021b).

2.37 React

React é uma biblioteca JavaScript para construção de interfaces de usuário. Ela é open-source, e foi criada pelo Facebook. A maior vantagem do React é poder construir componentes reutilizáveis, permitindo criar interfaces rapidamente e de forma eficiente. Mais informações podem ser vistas em (JS, 2021d).

2.38 React-bootstrap

React-bootstrap é uma biblioteca com os componentes visuais do Bootstrap construídos em React. Mais informações podem ser vistas em (BOOTSTRAP, 2021b).

2.39 React-confirm-alert

React-confirm-alert contém apenas um componente visual de alerta desenvolvido em React. Ele provê uma interface mais amigável para realizar um *bind* entre a lógica de negócio e a apresentação deste alerta. Mais informações podem ser vistas em (GA-MO, 2021).

2.40 React-google-recaptcha

React-google-recaptcha é um componente React que facilita a integração com o serviço Google reCaptcha. Ele provê uma interface amigável tanto para renderizar o componente de reCAPTCHA quanto para obter o token de validação através da API da Google. Mais informações podem ser vistas em (DOZOISCH, 2021).

2.41 React-share

React-share é um componente com um conjunto de botões React para compartilhamento em redes sociais. Ele fornece uma interface única para compartilhamento em diversas redes sociais como Facebook, LinkedIn, WhatsApp, Reddit, entre outras. Mais informações podem ser vistas em (NYGARDK, 2021).

2.42 APIs RESTful

REST (***RE****presentational* ***S****tate* ***T****ransfer*) é um estilo de arquitetura para sistemas distribuídos de hipermídia. Uma API web que conforma com o REST é chamada de API RESTful. Este estilo de arquitetura é aplicado sobre HTTP 2.20.

REST lida com **recursos**. Qualquer informação que consigamos nomear pode se tornar um recurso, como um documento, imagem, um serviço temporal, ou até um conjunto de outros recursos.

Um ***endpoint*** é uma função disponível através de uma API RESTful. Essa função pode ser algo como buscar um índice, atualizar uma publicação ou deletar um comentário. *Endpoints* efetuam uma tarefa específica, recebendo alguns parâmetros e retornando dados ao cliente.

Uma **rota** é o "nome" usado para acessar *endpoints*, e faz parte da URL. Uma rota pode possuir múltiplos *endpoints* associados a ela, variando de acordo com o verbo HTTP (e.g., GET, PUT, DELETE).

Mais informações sobre REST podem ser vistas em (BLUEHOST, 2021; FIELDING, 2021).

2.43 Sass

Sass é uma extensão do CSS que aumenta o poder de expressão da linguagem, incluindo elementos como variáveis, regras encadeadas, entre outros. Mais informações podem ser vistas em (COMUNITY, 2021).

2.44 SHA-256

SHA-256 é um algoritmo de *hashing* disponibilizado na RFC-6234. Ele foi projetado pela *United States National Security Agency* e publicado inicialmente em 2001. O 256 representa o tamanho em bits do *digest* usado no hashing. Mais informações podem ser vistas na própria RFC em (IETF, 2021).

2.45 Styled-components

Styled-components é uma biblioteca que permite escrever código CSS diretamente nos componentes React. Além de outras vantagens, essa abordagem facilita a manutenção ao centralizar os estilos diretamente nos componentes que os usam. Mais informações podem ser vistas em (COMMUNITY, 2021b).

2.46 Uniform Resource Identifier (URI)

Uniform Resource Identifier (URI) é uma sequência única de caracteres que identificam um recurso lógico ou físico. No contexto de requisições HTTP, cada recurso é identificado por um URI que é usado através do HTTP para identificá-lo. Como exemplo, uma URL é um tipo de URI. Mais informações podem ser vistas em (MOZILLA, 2021c).

2.47 Wireframe

Segundo (UNGER; CHANDLER, 2009), um ***wireframe*** é uma forma de identificar o conteúdo e a estrutura proposta, assim como os comportamentos funcionais, de uma exibição de página na internet ou de uma aplicação. Eles são, geralmente, apresentados

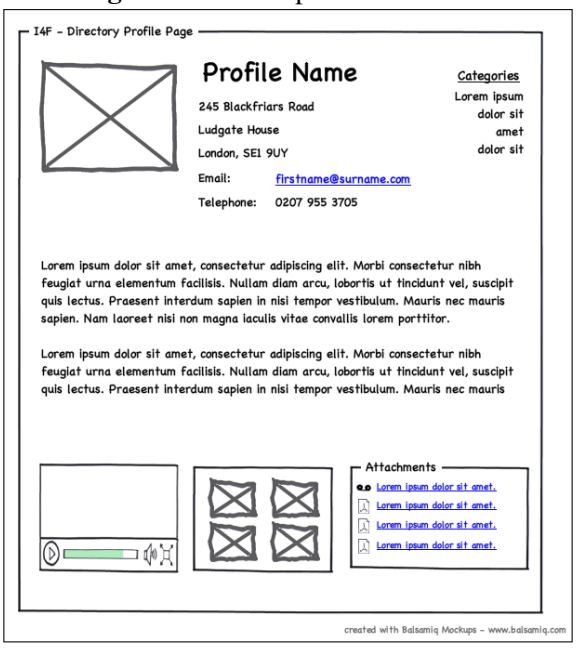
em escala de cinza, privados de elementos gráficos ou de conteúdo finalizado; em vez disso, eles usam conteúdo substituível para destacar apenas os locais representativos que podem ser usados como orientação no design visual.

Basicamente o *wireframe* é um protótipo de baixa fidelidade da tela. Ele é útil para identificar elementos como:

- Navegação;
- Seções de conteúdo;
- Necessidades de imagem e/ou de mídia;
- Elementos de forma;
- Chamadas para ações (CTAs).

O público para um *wireframe* varia de projeto para projeto, podendo ser uma única pessoa ou qualquer combinação de vários grupos. Alguns exemplos de públicos são analistas de negócio, gerentes de projeto, designers, desenvolvedores, usuários, entre outros. A Figura 2.3 mostra um exemplo de *wireframe*.

Figura 2.3: Exemplo de wireframe



Fonte: Wikipédia (WIKIPEDIA, 2009).

3 TRABALHOS RELACIONADOS

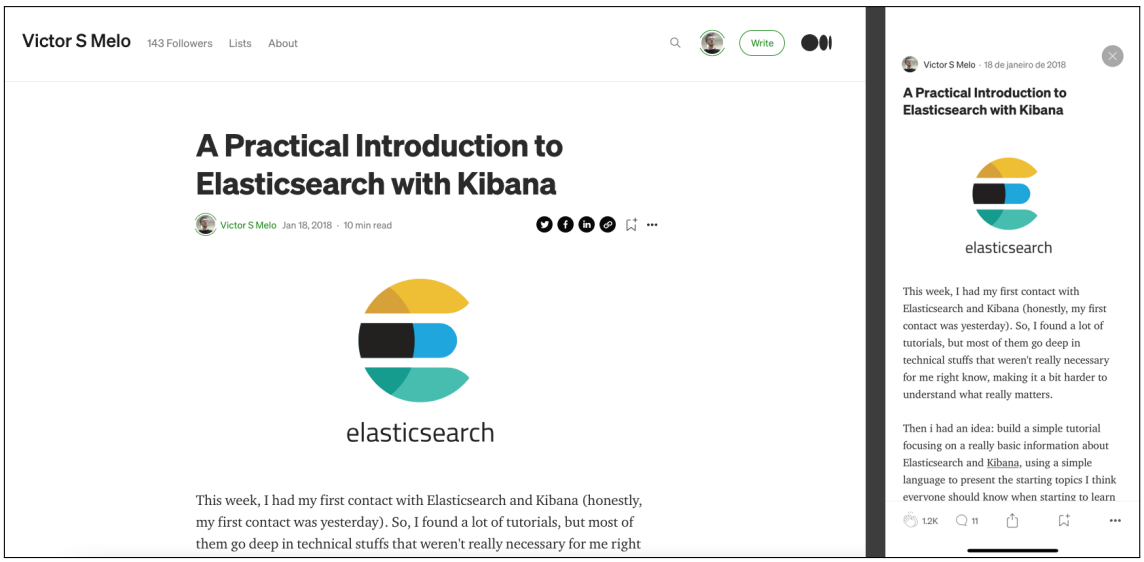
É difícil listar exaustivamente todos os trabalhos que serviram como inspiração para este projeto. Porém, existem três trabalhos que juntos sintetizam as características almejadas para ele. As seções abaixo discorrem sobre esses trabalhos relacionados.

3.1 Medium

O primeiro trabalho relacionado aqui é o **Medium**. Esta é uma plataforma aberta onde usuários podem criar e consumir publicações em texto. O Medium possui uma interface minimalista e intuitiva, exibida na Figura 3.1.

Das funcionalidades mais interessantes no Medium temos: as **`palmas`**, que indicam ao autor o quanto as pessoas gostaram de uma publicação; a tela de **`rascunho`**, que permite ao autor escrever a publicação sem necessariamente publicá-la; por fim vale citar os **`comentários`**, que são muito usados no Medium.

Figura 3.1: Interface do Medium no navegador e aplicativo iOS



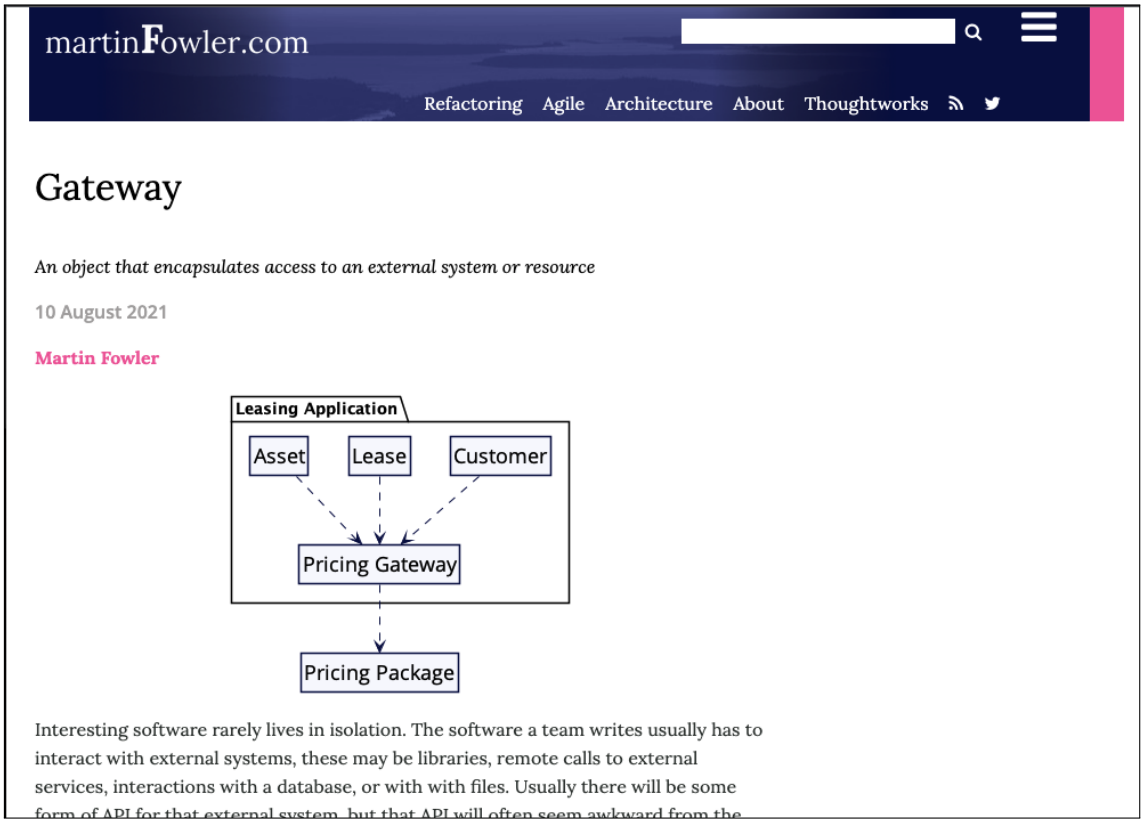
O projeto aqui apresentado não teve nenhuma intenção de ser uma plataforma. Porém, a interface minimalista do Medium serviu de inspiração na construção das telas, buscando direcionar e manter o foco no conteúdo. A Seção 5.1 apresenta como essa interface foi construída.

3.2 Blog do Martin Fowler

O segundo trabalho relacionado é o **blog do Martin Fowler** (Figura 3.2). Nele, além de links para conteúdos dos livros publicados, existe uma seção com diversos artigos sobre engenharia de software escritos pelo próprio Martin Fowler ou por outras pessoas da área. O site apresenta uma interface mais datada, porém o conteúdo publicado é de altíssima qualidade e profundidade. Existem publicações desde 1996, sendo um repositório imenso de conhecimento.

O aspecto mais interessante nos artigos do blog é que apesar de sua profundidade, eles são apresentados de uma forma tão didática que a leitura se torna fácil até pra quem é leigo. Essa didática se dá muitas vezes pelo uso consistente de palavras, referências externas e uma apresentação *top-down* dos temas.

Figura 3.2: Artigo sobre gateway no site do Martin Fowler



O projeto aqui apresentado se inspira no blog do Martin Fowler pela didática e profundidade dos temas discutidos. Longe de querer se tornar um blog similar, que só é possível após décadas de experiência, meu objetivo é tentar usar uma abordagem similar para apresentação dos conteúdos: escrever um texto profundo mas com uma abordagem *top-down*; adicionar referências externas sempre que facilitar o entendimento; e usar as

palavras de forma consistente.

3.3 Blog do David Walsh

O último projeto relacionado aqui é o **blog do David Walsh** (Figura 3.3). Este é um blog pessoal de um engenheiro de software sênior que trabalhou por 8 anos na Mozilla. Nele são publicados conteúdos sem formato ou tema específico. Como exemplos, ele contém textos mais teóricos, tutoriais longos e curtos, demos e até conteúdo relacionado à carreira e jornada do autor como engenheiro de software. O blog possui uma seção de comentários no final de cada publicação que é bastante ativa.

O projeto aqui desenvolvido se inspira no blog de David Walsh pelo aspecto caótico, porém eficiente de suas publicações. Esse caos se dá tanto pela variedade de temas quanto pelo formato das publicações. Tendo um foco em compartilhar aprendizados e experiência, essa flexibilidade é essencial para estimular a criatividade.

Figura 3.3: Blog do David Walsh

3.4 Resumo

Para resumir as seções acima, os principais aspectos que serviram de inspiração para este projeto foram:

- A interface minimalista do Medium;
- A didática e profundidade das publicações no blog do Martin Fowler;
- A liberdade sobre temas e formatos do blog do David Walsh.

Vale ressaltar que os dois últimos pontos não são contraditórios: é possível publicar conteúdo didático e profundo sem precisar pra isso restringir os temas ou formato das publicações.

4 VISÃO GERAL

Este capítulo visa descrever a metodologia de trabalho aplicada. Um planejamento foi feito antes do início da implementação, e revisado periodicamente durante ela. Isso é extremamente importante para direcionar bem o trabalho, tendo um objetivo norteador claro e alcançável. As seções abaixo discorrem sobre esse planejamento.

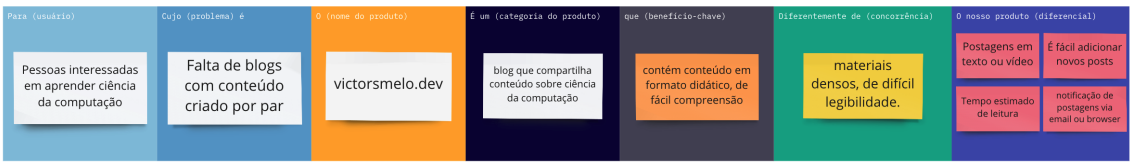
4.1 MIRO e o Lean Inception

A primeira ação feita para o desenvolvimento do blog foi um *kick-off* do projeto. O objetivo era transformar a visão de alto nível do blog em funcionalidades que poderiam ser desenvolvidas. Para isso foi usada a ferramenta MIRO (*www.miro.com*) com uma abordagem inspirada no *Lean Inception* (CAROLI, 2019). Essa abordagem se dividiu nas etapas abaixo:

1. Visão do produto.
2. Quadro "é - não é / faz - não faz"
3. Objetivo do produto
4. Personas e Jornadas
5. Brainstorm de funcionalidades
6. Revisão técnica, de negócio e de UX
7. Sequenciador

4.1.1 Visão do Produto

Figura 4.1: Inception - Visão do Produto



A Figura 4.1 apresenta a visão do produto. Ela ajuda a definir a essência do produto, e deve ser uma mensagem clara e convincente sobre ele. Sintetizando o resultado obtido dessa etapa, temos:

- **Usuário:** pessoas interessadas em aprender ciência da computação;
- **Problema:** falta de blogs com conteúdos criados por pares;
- **Nome do produto:** victorsmelo.dev;
- **Categoria do produto:** blog que compartilha sobre ciência da computação;
- **Benefício-chave:** contém conteúdo em formato didático, de fácil compreensão;
- **Problema da concorrência:** materiais densos, de difícil legibilidade;
- **Diferencial do produto:** postagens em texto ou vídeo; tempo estimado de leitura; notificação de postagens via e-mail ou browser; é fácil adicionar novos posts.

Apenas com essa descrição conseguimos identificar que trata-se de um blog, que o formato de conteúdo visa a didática, e que busca trazer uma visão de um par e não de um especialista. Esse último ponto pode ser visto tanto como uma vantagem quanto uma desvantagem. Porém, para o propósito de didática do blog, a visão de um par é importante já que teoricamente ele estaria mais próximo das dificuldades do leitor, identificando-as mais facilmente e apresentando o conteúdo da forma mais empática.

4.1.2 Quadro É - Não É / Faz - Não Faz

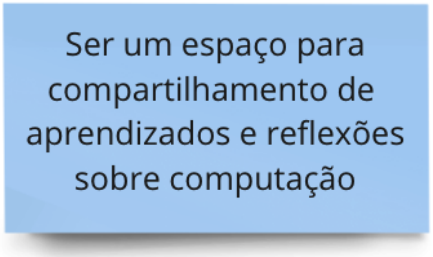
A Figura 4.2 apresenta o quadro. Muitas vezes é mais fácil descrever o que uma coisa não é ou não faz. Esse quadro busca classificar o produto de acordo com quatro diretrizes, indagando aspectos positivos e negativos. Pelo resultado percebe-se que o propósito do blog tá muito mais direcionado ao compartilhamento de experiência e aprendizados do que em ser uma ferramenta de ensino com postagens regulares.

4.1.3 Objetivo do Produto

A Figura 4.3 apresenta o objetivo do produto. Numa *lean inception* ela serviria para garantir que todos participantes compartilhem o mesmo entendimento do produto. Porém, como o projeto aqui é desenvolvido apenas por uma pessoa, ele serve como uma frase que sintetiza os passos anteriores. Nela lemos que o objetivo do blog é ser um espaço para compartilhamento de aprendizados e reflexões sobre computação.

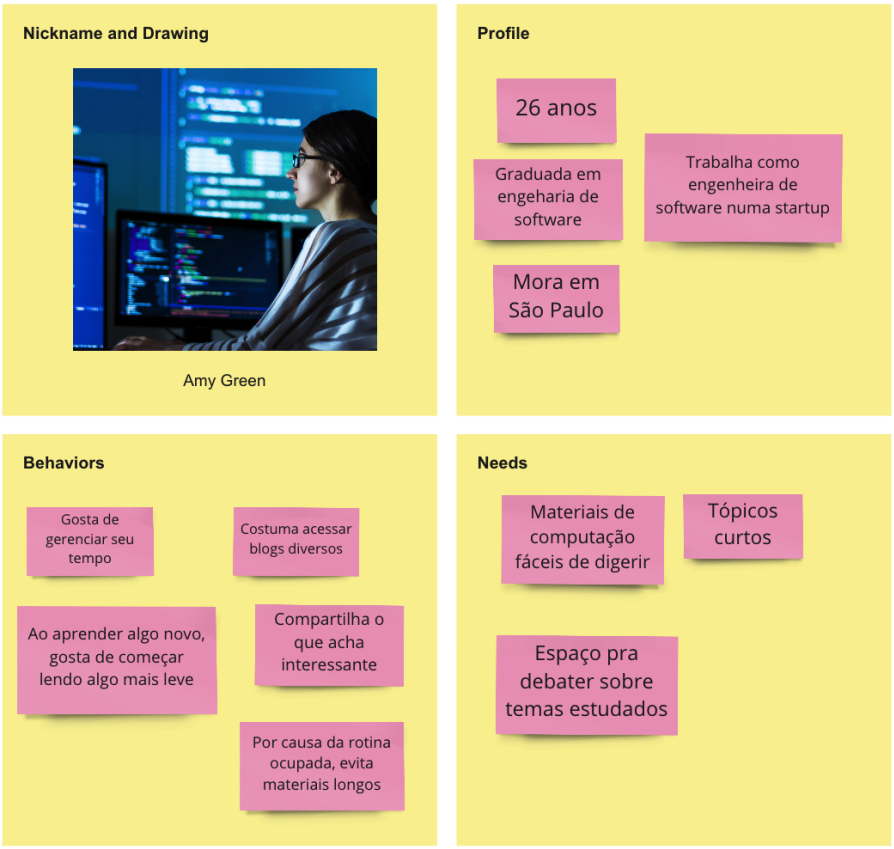
Figura 4.2: Inception - Tabela É - Não é / Faz - Não faz

Figura 4.3: Inception - Objetivo do Produto



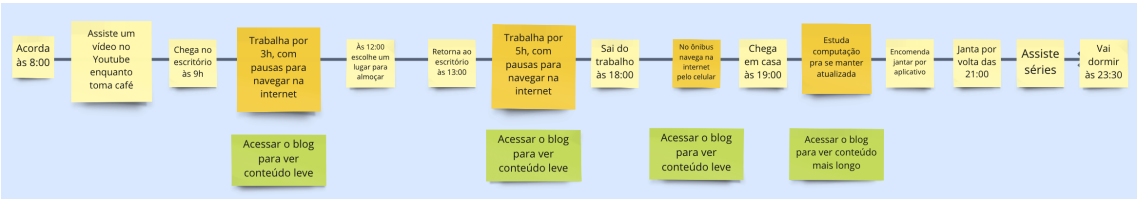
4.1.4 Personas e Jornadas

Figura 4.4: Inception - Persona Leitora



Definir personas é uma ação muito comum quando tentamos mapear usuários de um produto. A persona representa uma categoria de usuário, descrevendo seu papel e suas necessidades específicas. Ela ajuda na definição de funcionalidades a partir das necessidades do usuário final. Para este blog há duas categorias: o usuário **leitor** e o **escritor**.

Figura 4.5: Inception - Jornada da Persona Leitora



O **usuário leitor** é representado pela persona da Figura 4.4, que possui a jornada da Figura 4.5. Já o **usuário escritor** é visto na Figura 4.6, com a jornada da Figura 4.7. Em ambas jornadas estão destacados momentos-chave onde o blog pode ser integrado de alguma forma à rotina das personas.

Figura 4.6: Inception - Persona Escritora

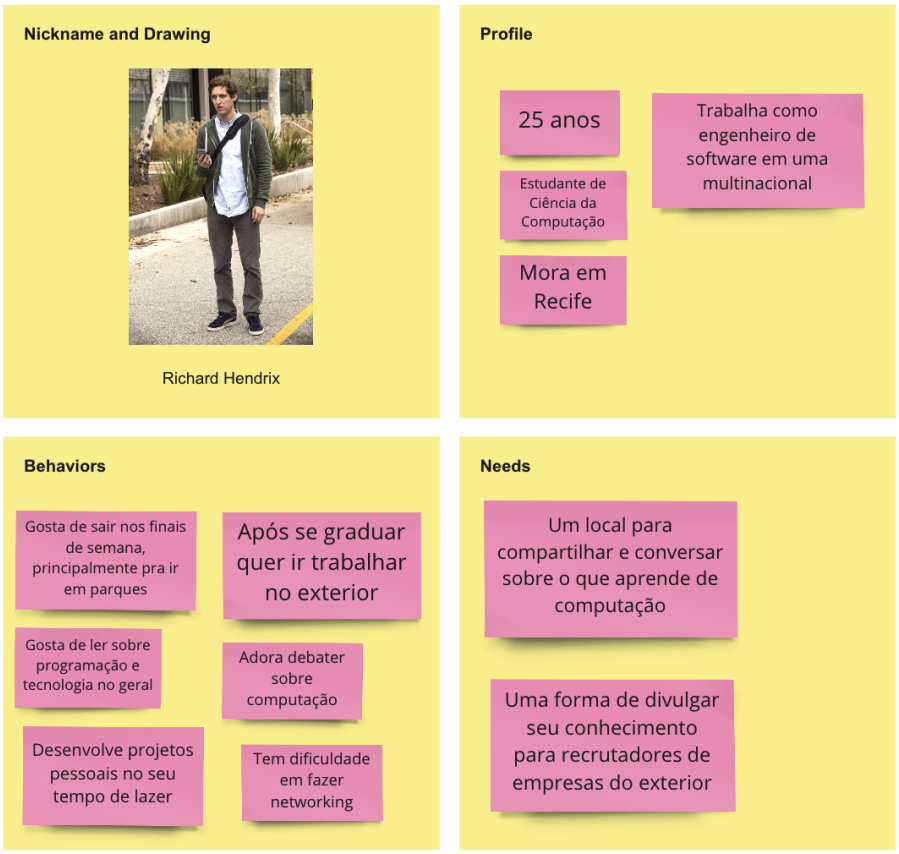
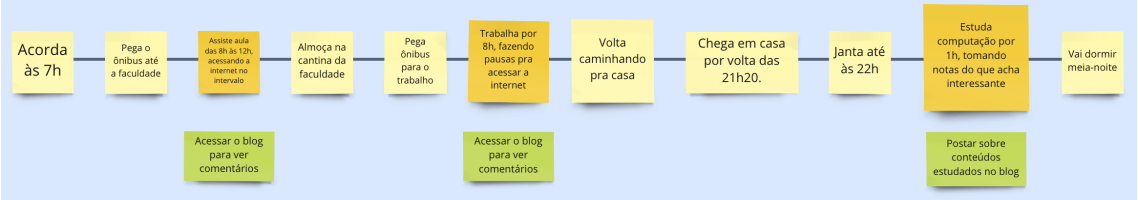


Figura 4.7: Inception - Jornada da Persona Escritora



4.1.5 Brainstorm de Funcionalidades

Nessa etapa considera-se as informações obtidas nas etapas anteriores para realizar um *brainstorm* de possíveis funcionalidades. Essas funcionalidades devem ser pensadas de forma a atender alguma necessidade do usuário, e devem ser descritas da forma mais simples possível.

A Figura 4.8 apresenta o resultado obtido nessa etapa, onde as funcionalidades foram agrupadas em: **features leitor**, englobando todas funcionalidades úteis ao leitor do blog; **design**, agrupando funcionalidades relacionadas ao projeto da interface; **features admin**, com as funcionalidades de administração do blog; e por fim **continuous deployment + infra**, contendo questões de entrega contínua e infraestrutura. Essas últimas, ainda que não sejam funcionalidades aos olhos do usuário final, foram adicionadas aqui por serem importantes para a manutenibilidade do sistema.

Figura 4.8: Inception - Brainstorm de Funcionalidades

4.1.6 Revisão Técnica, de Negócio e de UX

Essa etapa é essencial para priorização das funcionalidades geradas no *brainstorm*. Aqui elas são detalhadas, reavaliadas, normalizadas e algumas até descartadas.

Cada funcionalidade é avaliada de acordo com sua importância e o nível de confiança do time sobre o que é essa funcionalidade e como ela pode ser implementada. Para **esforço**, **negócio** e de **UX**, classificamos as funcionalidades com uma marcação que vai de um a três, de acordo com a importância dela para cada critério. Além disso, avaliamos a **incerteza de negócio** (*business agreement*), que mede o quão clara é a descrição da funcionalidade para o time, e a **incerteza técnica** (*technical certainty*), que mede o quanto o time sabe como implementar a funcionalidade.

A Figura 4.9 mostra o quadro que é usado para fazer essas medições, enquanto a Figura 4.10 mostra o resultado dessa etapa, contendo os cartões gerados.

Figura 4.9: Inception - Modelo de Revisão Técnica, de Negócio e de UX

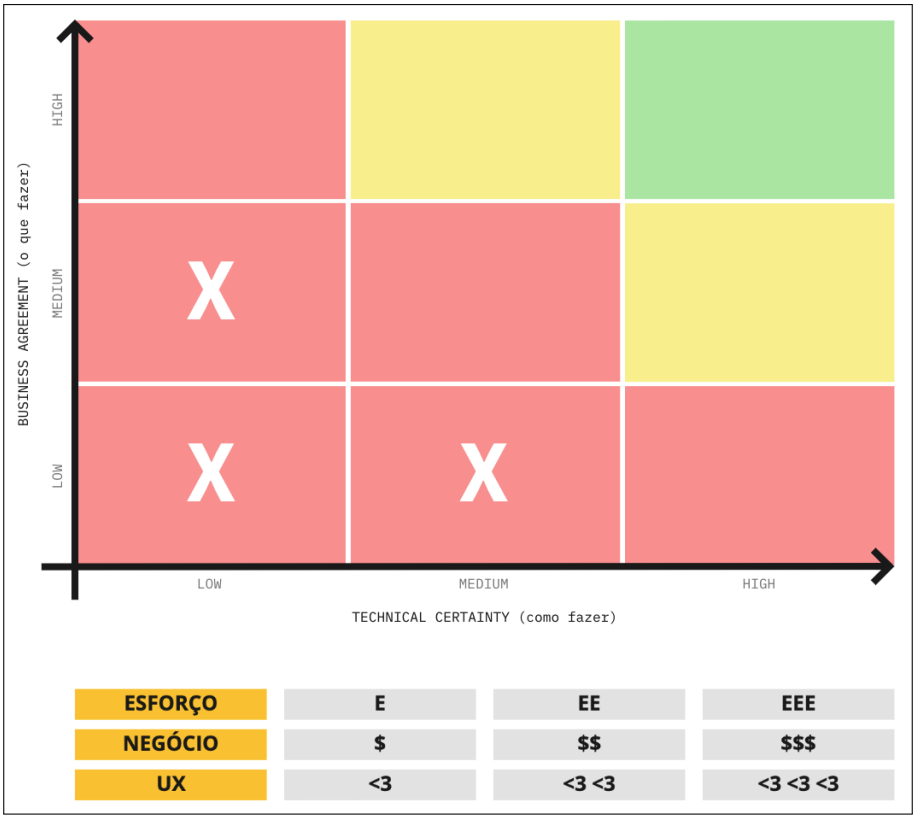


Figura 4.10: Inception - Resultado da Revisão Técnica, de Negócio e de UX

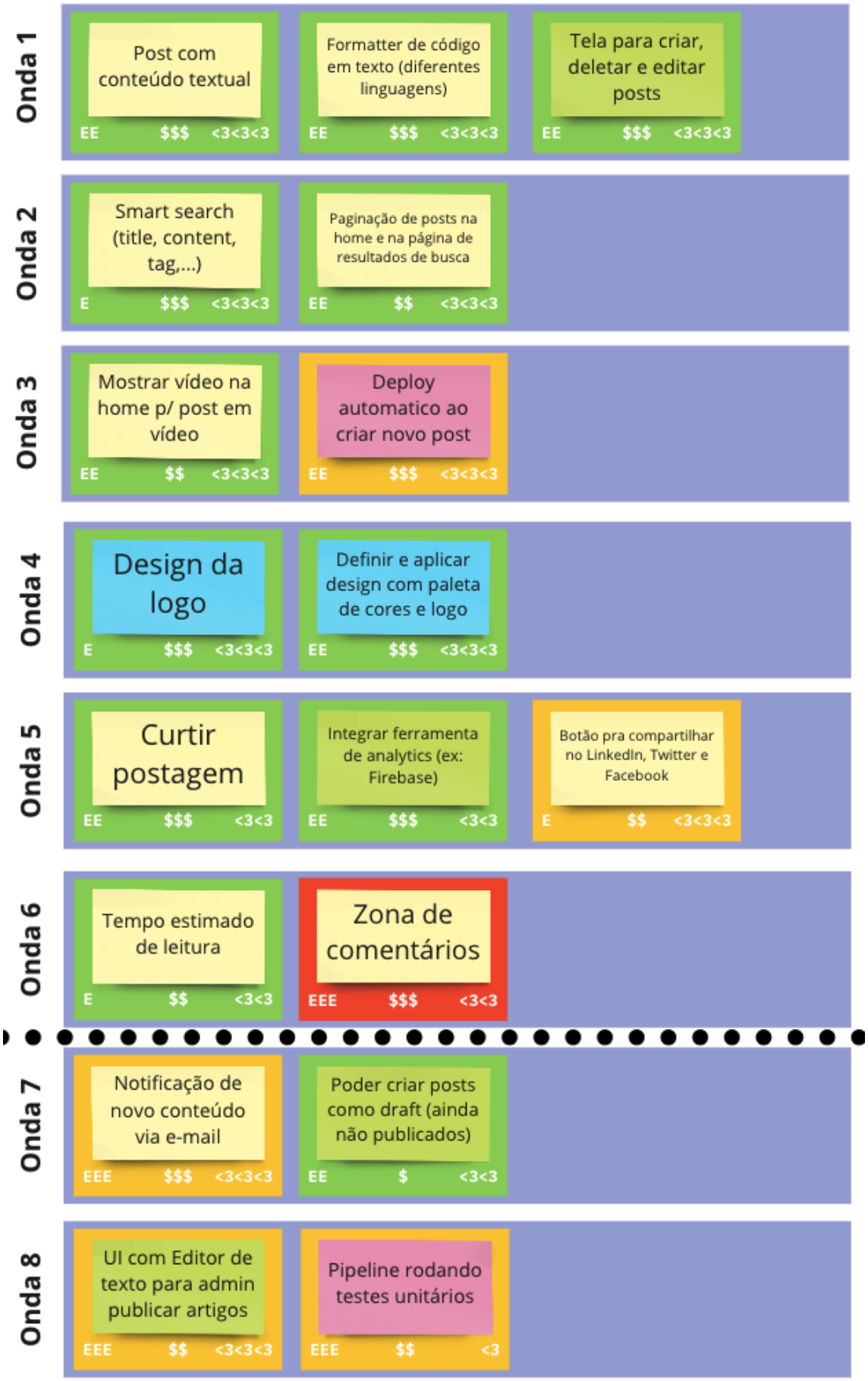
4.1.7 Sequenciador

O sequenciador foi a última etapa aplicada. Nele sequenciamos o desenvolvimento das funcionalidades em ondas, similar às *sprints* do Scrum. A diferença em relação às *sprints* está na forma com que as ondas devem ser definidas:

- Uma onda pode conter, no máximo, três cartões;
- Uma onda não pode conter mais de um cartão vermelho;
- Uma onda não pode conter três cartões somente amarelos ou vermelhos;
- A soma de esforço dos cartões não pode ultrapassar cinco “E”;
- A soma de valor dos cartões não pode ser menor de quatro “$” e quatro corações;
- Se um cartão depende de outro, esse outro deve estar em alguma onda anterior.

O resultado final de todo esse processo foi a sequência visível na Figura 4.11. As ondas após a linha tracejada foram definidas como opcionais, e realmente acabaram não sendo implementadas, enquanto as funcionalidades das ondas 1 a 6 foram implementadas total ou parcialmente.

Figura 4.11: Inception - Sequenciador



5 DESENVOLVIMENTO E IMPLEMENTAÇÃO

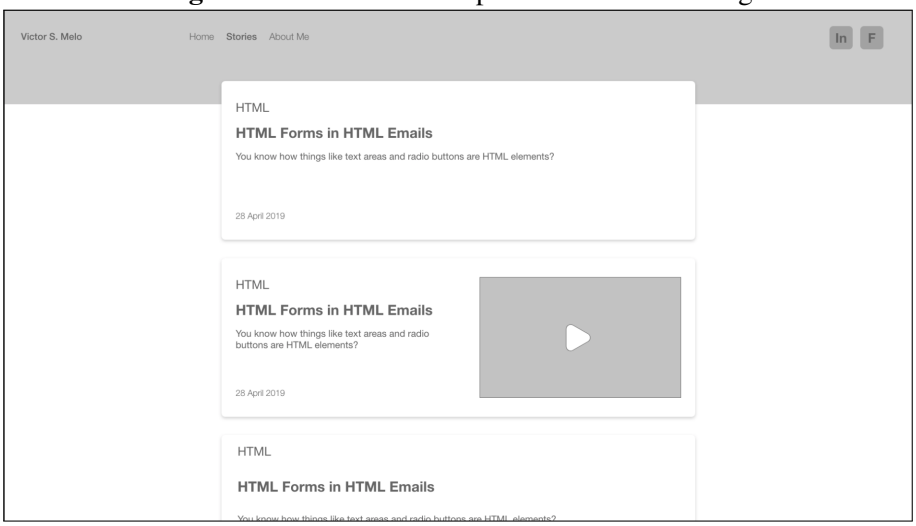
Este capítulo explica o resultado final do trabalho a partir de diferentes perspectivas. Para isso foi usada uma abordagem *top-down*: a Seção 5.1 começa analisando a interface do blog, mostrando a visão do usuário final sobre o sistema; já a Seção 5.2 apresenta uma visão geral da arquitetura do sistema; na Seção 5.3 são exploradas as funcionalidades implementadas; já as Seções 5.5 e 5.6 discorrem sobre os repositórios de código, visando mostrar como ele foi estruturado e quais dependências existem; já a Seção 5.7 discorre sobre a infraestrutura; por fim a Seção 5.8 mostra alguns *scripts* implementados para automatização de tarefas recorrentes como subida do ambiente local ou *deploy* da aplicação.

5.1 Projeto de Design e Usabilidade

O design da interface do blog foi realizado em um processo iterativo que começou em 2020. Foram utilizadas ferramentas como Adobe XD e Sketch para desenvolver tanto os *wireframes* (Seção 2.47) quanto a logo.

A primeira versão do blog pode ser vista na Figura 5.1. No menu superior podemos ver que existe uma aba *stories*, que é onde ficariam as publicações. A *home* seria usada para outro propósito.

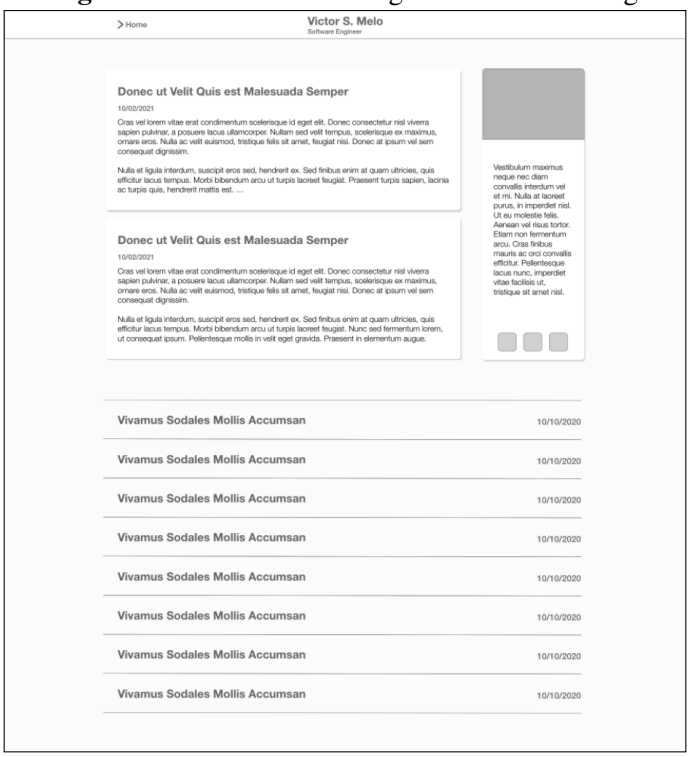
Figura 5.1: Wireframe da primeira versão do blog



Já na segunda versão (Figura 5.2, as publicações já ficam na `home`, porém estão destacadas apenas as duas mais recentes, enquanto as mais antigas exibem apenas seu título e data de criação. Nessa versão a barra lateral contém as informações que, na

versão final, pertencem à página `about me` (Seção 5.3.4).

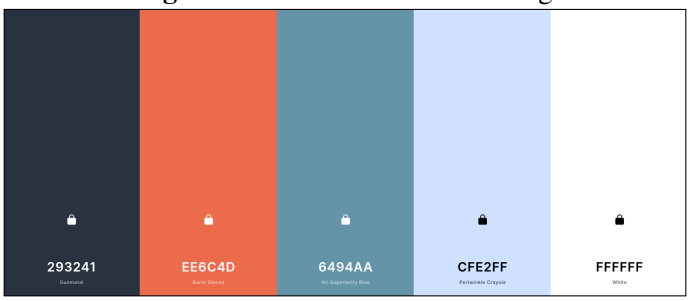
Figura 5.2: Wireframe da segunda versão do blog



A terceira versão de *wireframe* é a versão final. Essa é a única versão em que foram desenhados *wireframes* para todas as telas. Esses *wireframes* e os resultados finais que foram obtidos a partir deles são apresentados no Anexo A.

5.1.1 Cores

Figura 5.3: Paleta de cores do blog



Fonte: https://coolors.co

A Figura 5.3 apresenta a paleta de cores escolhida para o sistema. Quando pensamos na experiência do usuário, um bom uso das cores é essencial. Cada cor transmite uma mensagem, um sentimento, e é por isso existem livros incríveis, como o da Eva Heller (HELLER, 2014), que falam sobre como as cores influenciam a percepção humana.

Na Figura 5.4 vemos algumas cores, as emoções associadas a elas e as marcas que as usam.

Para o blog foi escolhido como cor predominante o **branco**, por ser uma cor neutra que não irá cansar a vista. Também foram escolhidos **três tons de azul** como cores secundárias, buscando transmitir segurança e confiança. A cor terciária escolhida foi um **tom avermelhado**, por ser complementar aos azuis, para destacar os elementos principais como botões importantes, palavras-chave ou até a própria logo do blog.

Figura 5.4: Guia emocional das cores

Fonte: (KIM, 2016)

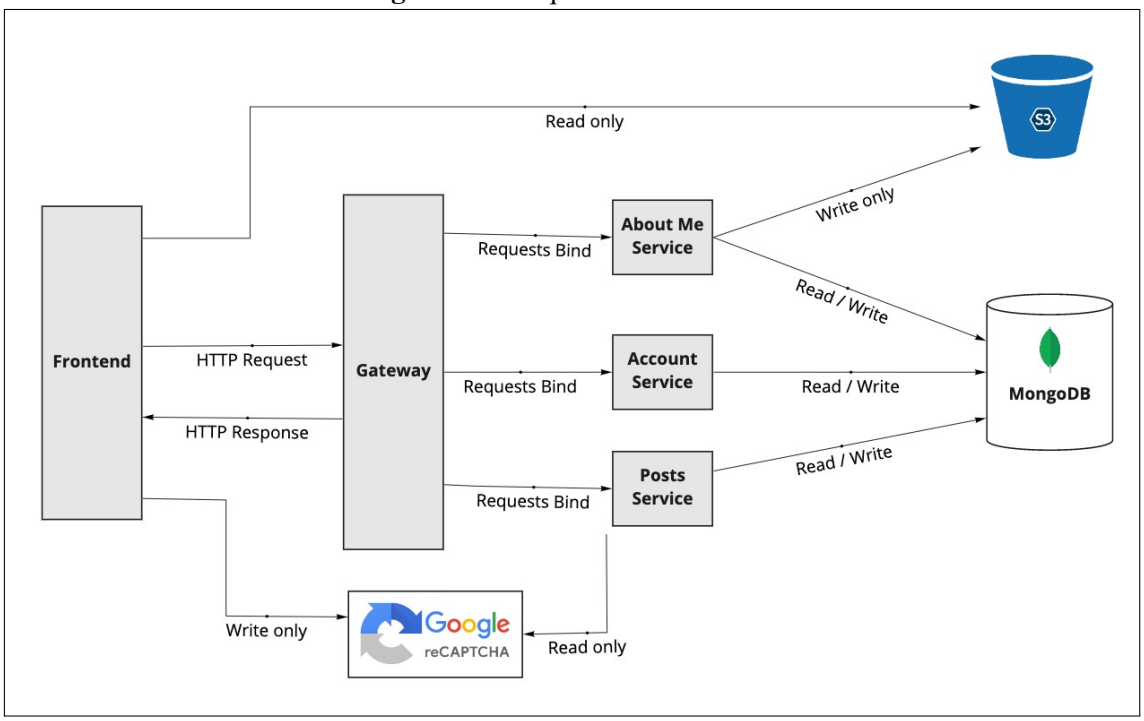
5.1.2 Logo

A Figura 5.5 apresenta a logo do blog. Ela foi projetada considerando as cores da paleta escolhida (Figura 5.3). A logo utiliza o tom avermelhado, complementar ao fundo azulado da barra superior do blog, e possui um trabalho de degradê com luz e sombra. Ela foi definida como uma composição de um **V** e um **M**, iniciais do autor do blog.

Figura 5.5: Logo do blog

5.2 Visão Geral da Arquitetura

Figura 5.6: Arquitetura do sistema



Na Figura 5.6 podemos ver a arquitetura completa do sistema. Nela temos o **Frontend** representando a página web acessada pelo usuário no navegador, enquanto todos os outros componentes são implementados no lado do servidor. É interessante perceber que, pela arquitetura escolhida, é possível no futuro haver outros *front-ends*, como um aplicativo mobile. Contanto que o cliente saiba fazer requisições HTTP, ele conseguirá se comunicar com o servidor. O **Gateway** é quem recebe as requisições e executa validações internas de segurança. Caso esteja tudo certo, ele encaminha a requisição para o devido serviço.

Os serviços **About Me**, **Account** e **Posts** são os componentes que efetivamente

operam sobre as requisições e retornam o resultado ao cliente. Esses componentes se comunicam com o banco de dados **MongoDB** tanto pra ler quanto pra escrever, dependendo da operação realizada. Esse banco armazena todos os dados do blog, já que o **bucket S3** é usado atualmente apenas para armazenar a imagem de perfil exibida na página *about me*. Por precisar alterar essa imagem de perfil, o About Me Service é o único que possui permissão de escrita no S3, enquanto o Frontend possui apenas permissão de leitura .

O Posts Service se comunica com o **Google reCaptcha** a fim de validar *tokens* enviados pelo cliente. Tanto o uso do bucket S3 quanto do reCAPTCHA são detalhados na Seção 5.3.

Sobre questões de segurança, vale comentar que o Gateway aceita apenas requisições HTTPS, por segurança. Se o usuário tentar utilizar HTTP puro, o servidor retornará status 301 (*permanent redirect*), enviando-o automaticamente para a URL com HTTPS.

Outra responsabilidade do Gateway é definir uma cadeia de funções que executarão sobre cada requisição, onde a última função dessa cadeia sempre é algum dos *services*. As funções intermediárias são chamadas *middlewares*, e são elas que executam as validações de segurança. Como exemplo, a etapa que verifica se o usuário está logado é implementada como um desses *middlewares*.

Sobre o **Google reCAPTCHA**, ele é usado na funcionalidade de comentários. No navegador, quando o usuário clica no verificador é enviado uma requisição ao servidor da Google, retornando um *token* dessa requisição caso seja válida. Ao criar um comentário, esse *token* é enviado ao servidor do blog, que vai validá-lo junto ao servidor da Google. Caso não esteja válido, o comentário será recusado. Caso seja, vai ser publicado e ficará visível à todos.

5.3 Funcionalidades

No Capítulo 4 foram mostradas diferentes funcionalidades planejadas para o blog, porém nem todas foram implementadas. Para entender o motivo disso, é interessante discorrer sobre o **triângulo da gerência de projetos**.

O *Project Management Book of Knowledge* PMBOK® defende que todo projeto é governado por três **restrições**: custo, escopo e tempo (WYNGAARD; PRETORIUS; PRETORIUS, 2012). Tais restrições são exemplificadas pelo triângulo na Figura 5.7.

Traduzindo essas restrições para o contexto do blog, podemos entendê-las como:

- **custo:** tempo dedicado ativamente no projeto;
- **tempo:** tempo total disponível;
- **escopo:** todo trabalho já realizado.

A área interna do triângulo representa a **qualidade** do projeto. Uma forma de ilustrar o progresso do projeto é imaginar que existem dois triângulos: um Δ e outro $\delta(t)$, onde t é um instante qualquer do tempo de vida do projeto. O triângulo Δ é apresentado na Figura 5.7, e representa a estimativa inicial do projeto. Ele se mantém fixo, sendo usado como referência. Já o triângulo $\delta(t)$ representa o progresso do projeto até o instante t. Seja $a(x)$ a área de um triângulo x, temos como propriedade $a(\delta(t_j)) > a(\delta(t_i))$ para todo $t_j > t_i$. Em outras palavras, a área do triângulo $\delta(t)$ só pode aumentar com o tempo. Chamaremos de **requisitos** os vértices desse triângulo $\delta(t)$.

Com essa representação gráfica é fácil perceber que a finalização ideal do projeto em um instante t_f é quando $\delta(t_f) = \Delta$, ou seja, os requisitos se igualam às restrições. Essa igualdade significa que a estimativa Δ foi 100% correta.

Figura 5.7: Expectativa de um projeto

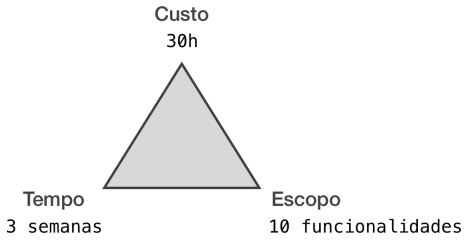
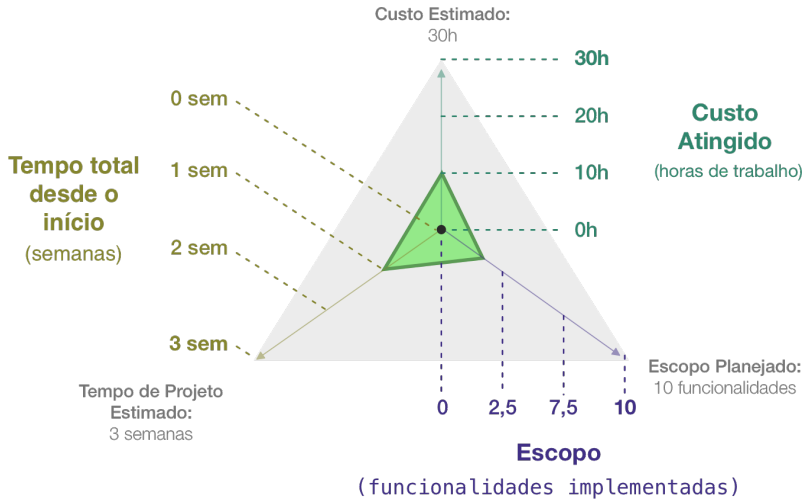


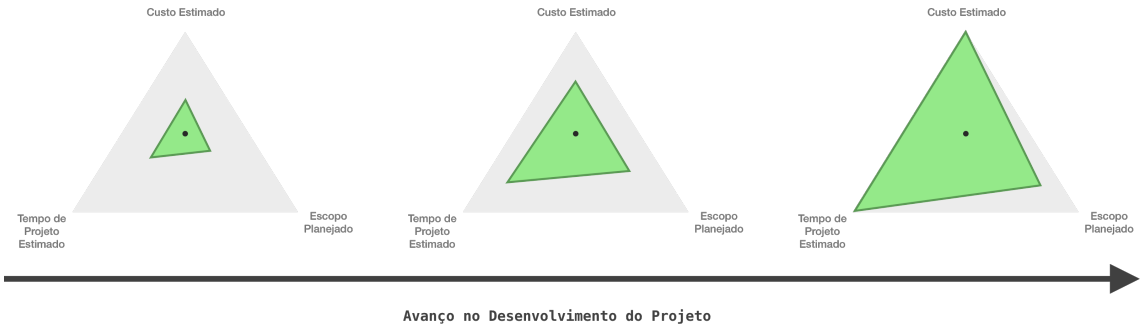
Figura 5.8: Triângulo do projeto realizado



A Figura 5.8 apresenta um exemplo da relação entre os triângulos descritos, en-

quanto a Figura 5.9 exemplifica a variação em $\delta(t)$ à medida em que o 1 t avança.

Figura 5.9: Avanço no desenvolvimento do projeto



Nos projetos sempre existem certas restrições que não podem ser alteradas, e no caso do projeto deste blog, o **tempo** é uma delas: não há como entregar o projeto após a data esperada pela universidade. Já o **custo** de tempo dedicado ao projeto pode até ser alterado, mas considerando que é um projeto individual, existe uma relação $custo \leq (tempo - \alpha)$, onde α representa o tempo dedicado às outras atividades essenciais não relacionadas ao projeto (e.g., dormir, trabalhar, comer).

A restrição mais facilmente manipulável acaba sendo o **escopo**, e isso explica porque há uma diferença entre as funcionalidades planejadas, apresentadas anteriormente, e as funcionalidades efetivamente desenvolvidas.

Dados os pontos acima, a lista de funcionalidades efetivamente implementadas podem ser agrupadas em diferentes seções: **Administração**, **Posts**, **Busca**, **Sobre Mim** e **Analytics**.

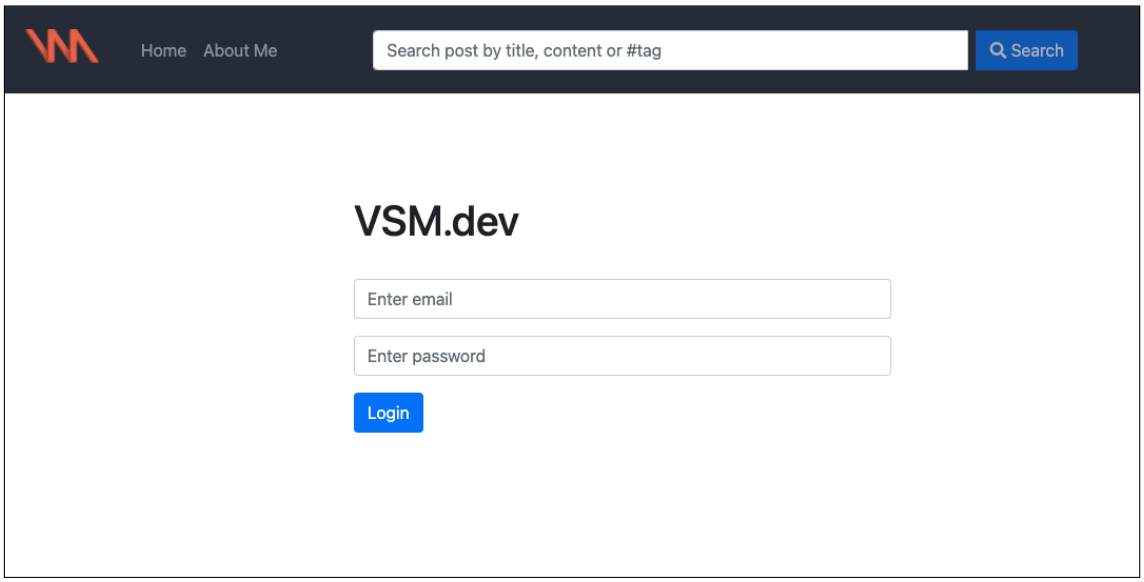
5.3.1 Administração

A seção de administração diz respeito a todas funcionalidades necessárias para se administrar o blog, incluindo adicionar, modificar e deletar conteúdo, além de moderar comentários nas publicações.

5.3.1.1 Login

A funcionalidade de login é essencial para administração. Neste blog existe apenas um tipo de usuário: o administrador. Desta forma não há um fluxo para criação de conta. Os dados do administrador são inseridos manualmente no banco de dados utili-

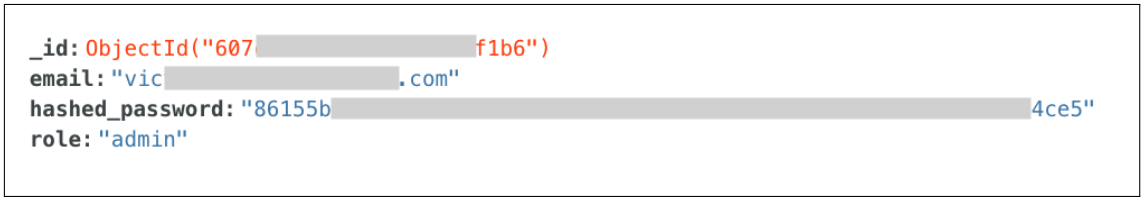
Figura 5.10: Tela de login



zando a interface provida pelo MongoDB Atlas (Seção 2.26).

A tela de login pode ser vista na Figura 5.10. Nela o administrador insere seu e-mail e senha, que já estão cadastrados no banco de dados conforme mostra a Figura 5.11. Os dados são transmitidos via HTTPS pra garantir a confidencialidade. No servidor, o e-mail e senha recebidos são comparados com os armazenados no banco, sendo que para isso a senha é cifrada usando SHA-256 (Seção 2.44).

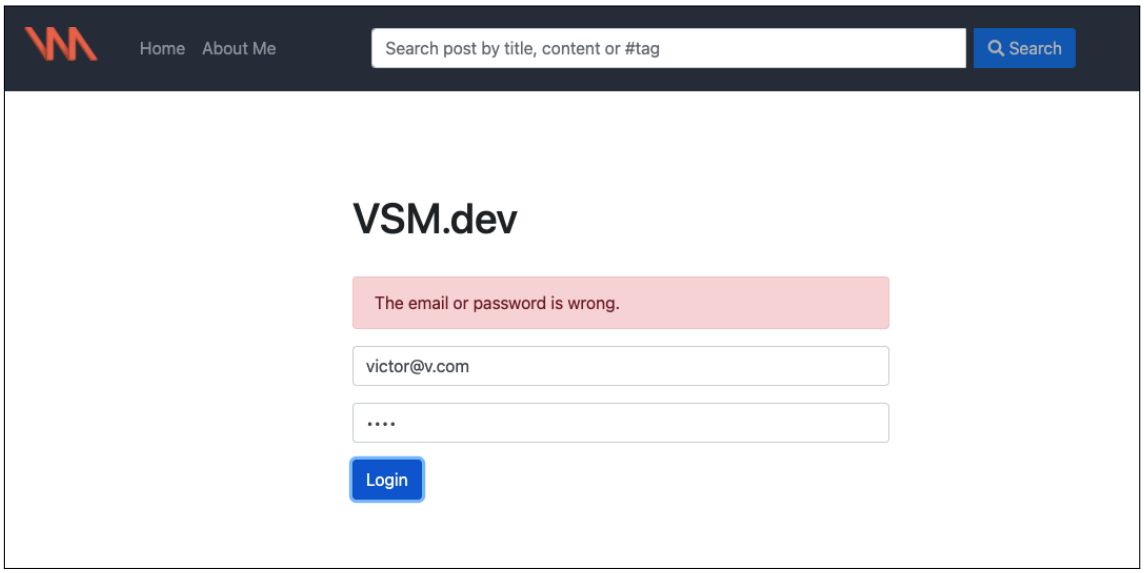
Figura 5.11: Dados de login no MongoDB



Caso os dados recebidos sejam inválidos, é exibida a mensagem de erro da Figura 5.12. Essa mensagem é propositalmente genérica para evitar que, em caso de tentativa de invasão, o invasor não saiba se errou o e-mail ou a senha.

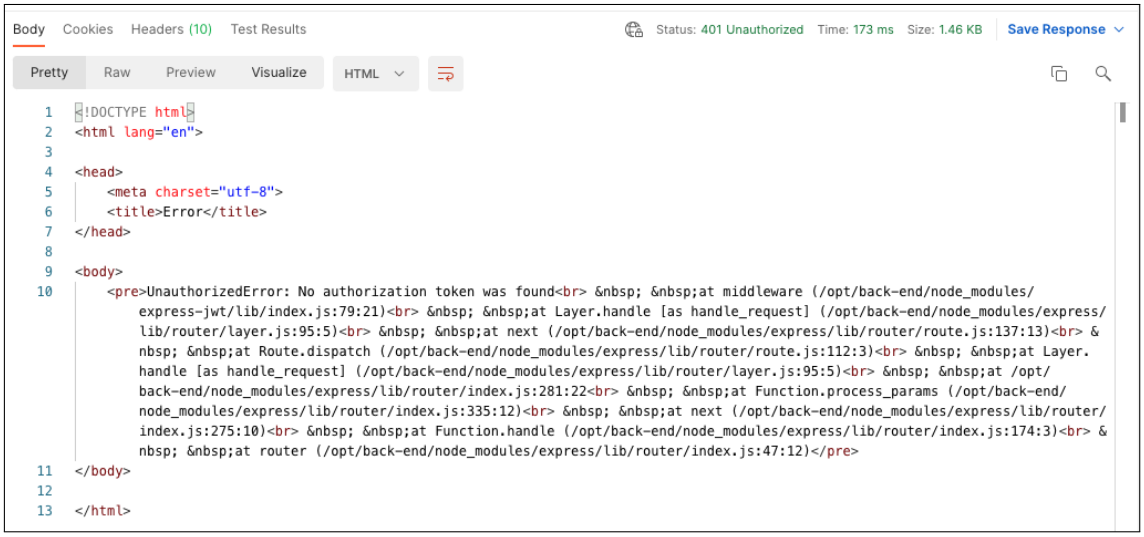
Em caso de sucesso, o servidor gera um *JSON Web Token*, conforme descrito na Seção 2.22, que é enviado ao cliente. O token é armazenado no cliente como um *cookie* que é enviado junto a cada nova requisição. Se essa requisição exigir login, o token é verificado. O token foi configurado para expirar após 24 horas, sendo necessário relogar para gerar um novo token depois desse período.

Figura 5.12: Mensagem de erro no login



Ao tentar executar qualquer operação que exija login sem token ou com token expirado, o servidor retornará erro com status 401 (*unauthorized*), conforme exemplificado na Figura 5.13. Porém, se a requisição não autorizada for uma tentativa de acesso à uma página de administração, como o *dashboard* (Seção 5.3.1.4), o usuário é apenas redirecionado para a página inicial do blog.

Figura 5.13: *Unauthorized* retornado do servidor ao tentar criar uma publicação sem token válido.

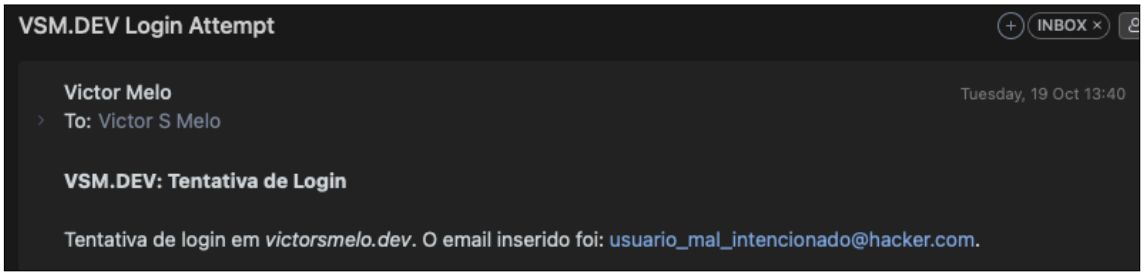


5.3.1.2 Notificação de Tentativa de Login

A única maneira de chegar à página de login é conhecendo o caminho na URL. Ainda assim, caso algum usuário mal intencionado consiga acessar essa página, por enquanto não existe nenhuma medida de segurança que limite a quantidade de tentativas. Enquanto essa funcionalidade não for implementada, como medida paliativa o servidor envia um e-mail de notificação sempre que alguém tenta realizar login, independente de ser um login com sucesso ou erro.

Para implementar esse serviço foi utilizado o Simple Email Service 2.3. Esse e-mail enviado serve como um alerta ao administrador do site, que ao identificar uma tentativa de invasão, pode tomar alguma ação como desativar o login temporariamente. A Figura 5.14 mostra o conteúdo desse e-mail. Pelo horário, quantidade de e-mails ou e-mail inserido, o administrador consegue saber se foi uma tentativa dele mesmo ou se foi outra pessoa. Foi evitado inserir qualquer informação identificável como IP, a fim de evitar o tráfego de informações que pode violar a Lei Geral de Proteção de Dados. A senha inserida também não é trafegada, por segurança.

Figura 5.14: Conteúdo do e-mail notificando tentativa de login



5.3.1.3 Logout

A funcionalidade de logout foi implementada de forma simples. Idealmente, gostaríamos que ao realizar o logout, o token fosse invalidado no lado do servidor e então removido do navegador. Porém, por restrição de tempo apenas a segunda etapa foi implementada. A vulnerabilidade dessa abordagem é que o *token*, desde que esteja no prazo de validade, pode ser salvo e usado posteriormente em outro contexto (e.g., outro navegador, outra máquina, outra pessoa usando). Normalmente isso é algo que não desejamos.

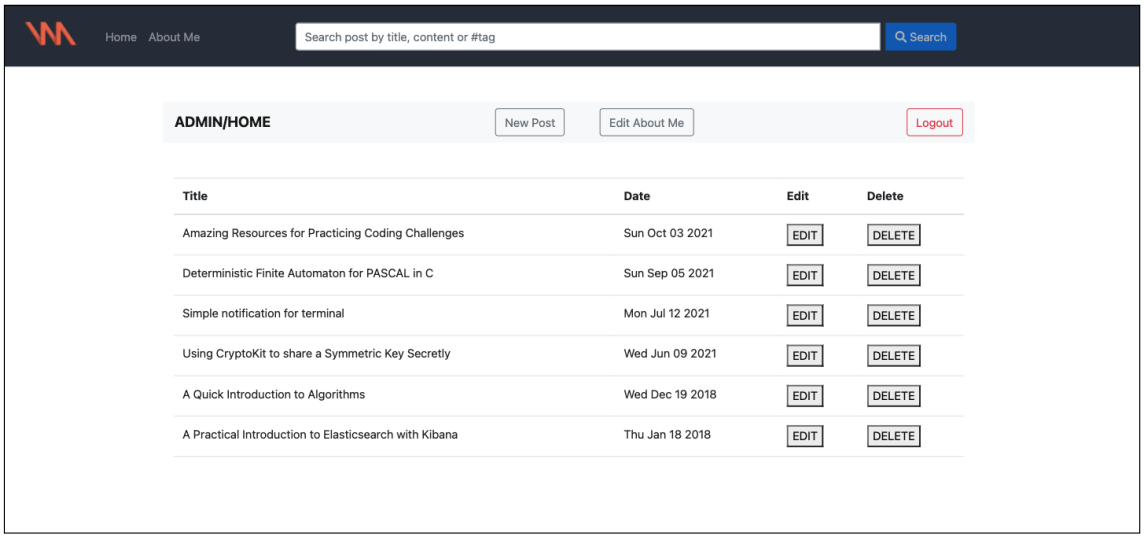
Assim, a funcionalidade de logout apenas detecta que o usuário clicou no botão de logout, apaga o cookie do token e então encaminha o usuário para a página de login. Esse

botão de logout fica no canto superior direito da página de *dashboard*, conforme mostra a Figura 5.15.

5.3.1.4 Dashboard

O *dashboard* (Figura 5.15) é onde o administrador gerencia o conteúdo do blog. Ele apresenta uma tabela com as publicações já criadas. Existem dois botões para operar sobre essas publicações, e tanto o comportamento do botão **EDIT** quanto do **DELETE** são descritos na Seção 5.3.2.7.

Figura 5.15: Dashboard de administração

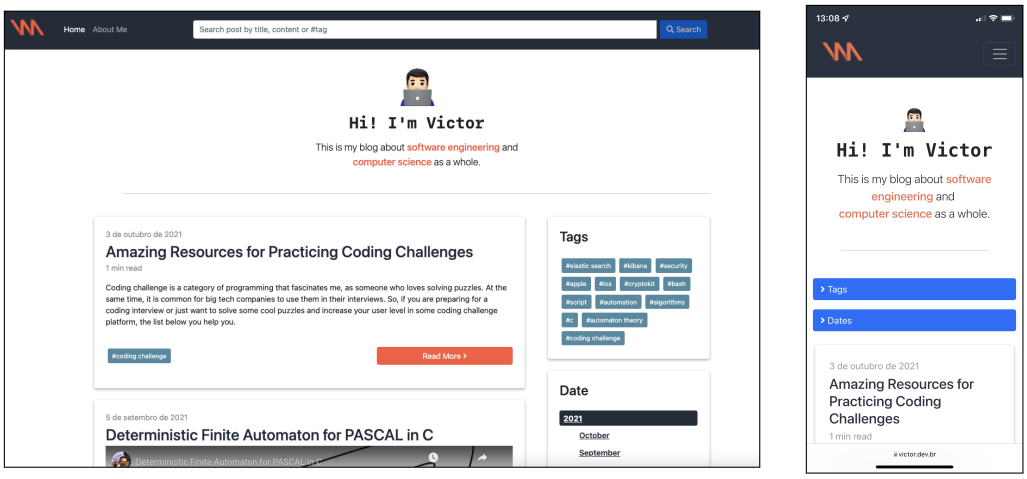


O botão **New Post** encaminha para a tela de criação de publicação (Seção 5.3.2.7). O botão **Edit About Me** encaminha para a tela de edição do conteúdo da página *About Me* (Figura 5.29). Já o botão de **Logout** foi descrito na Seção 5.3.1.3.

5.3.2 Posts

Esta seção apresenta as implementações relacionadas às publicações do blog. Esta é a seção que contém mais funcionalidades, dado que as publicações são elementos centrais em um blog.

Figura 5.16: Tela inicial em ambiente desktop e mobile



5.3.2.1 Exibição e Paginação na Tela Inicial

A Figura 5.16 mostra a página inicial do blog. Nela é exibida uma lista de publicações, ordenada cronologicamente da mais recente para a mais antiga. Cada publicação na *home* tem o intuito de despertar o interesse do leitor para ler o conteúdo completo. Assim, cada publicação é estruturada com os seguintes elementos:

- A **data** da publicação, para o leitor saber se o conteúdo é recente;
- O **título** da publicação;
- O **tempo estimado de leitura**, para o usuário saber se terá tempo de ler;
- O **resumo** do conteúdo publicado;
- As **tags**, mostrando quais temas são abordados;
- Um botão **read more** deixando claro ao usuário onde clicar.

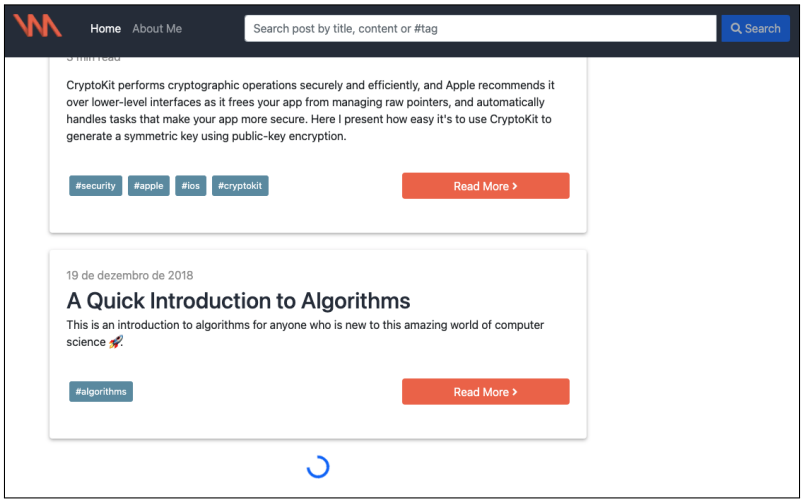
Vale destacar que além do botão **read more**, o título também é clicável. Isso foi feito visando uma melhor usabilidade, dado que título clicável é um padrão em diferente sistemas web.

Por escalabilidade, foi usada uma abordagem de paginação para exibir as publicações. Dessa forma, a tela inicial carrega as N últimas publicações, e esse valor N é indicado na requisição ao servidor. Para paginar, foi escolhida uma **abordagem baseada em *scroll***, ao invés de botões de páginas. Assim, ao chegar no final da página, as próximas publicações são carregadas. Isso foi feito inspirado pelo comportamento de diferentes sistemas atuais como Youtube, Facebook ou Instagram, além dela ser mais intuitiva para ambientes mobile.

Em um estudo publicado na revista *Behaviour & Information Technology* em 2004

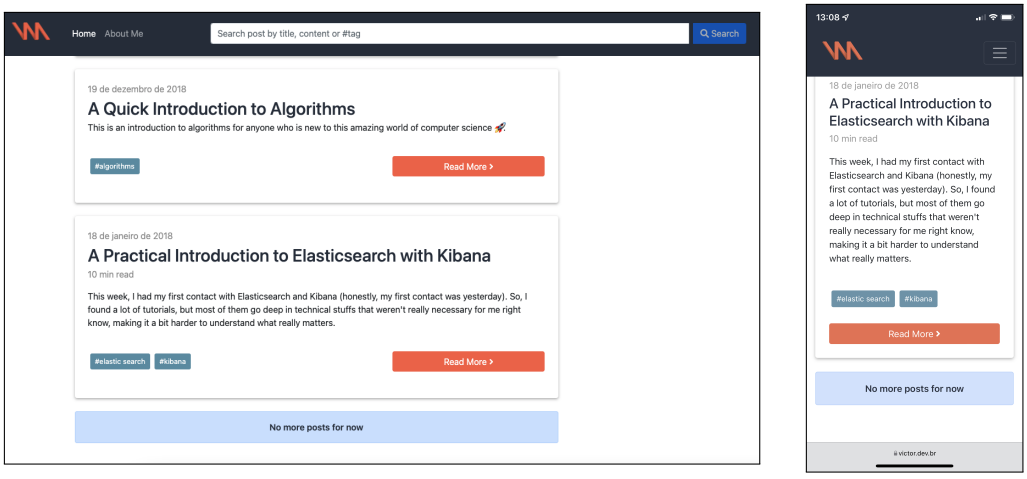
(NAH, 2004), foi constatado que ao usar elementos visuais indicadores de progresso, o tempo de espera é percebido pelo usuário como mais curto do que efetivamente é. Por conta disso foi implementado um *spinner*, que é exibido enquanto uma página está sendo carregada. A Figura 5.17 exibe esse *spinner*.

Figura 5.17: Spinner de paginação



Ao carregar todas as publicações existentes, é exibida uma mensagem indicadora, e assim não é mais possível buscar novas páginas do servidor. A Figura 5.18 exibe esse comportamento.

Figura 5.18: Final da paginação na tela inicial



5.3.2.2 Tempo Estimado de Leitura

A funcionalidade de tempo estimado de leitura foi trivial de ser desenvolvida, sendo usada apenas a seguinte equação:

$$\texttt{reading\_time(post)} = \lfloor \frac{\texttt{words\_count}}{250} \rceil \tag{5.1}$$

onde `words_count` é o número de palavras do `post`, e o denominador 250 é a estimativa de palavras lidas por minuto por um adulto padrão. O resultado dessa divisão é arredondado para o valor inteiro mais próximo, que indica o tempo de leitura em minutos.

Esse valor 250 foi obtido a partir de duas fontes: a primeira foi um artigo online sobre como essa estimativa de tempo de leitura aumenta o engajamento do usuário (HOLLAND, 2014); a segunda foi o algoritmo aplicado no Medium (MEDIUM, 2012), onde foi feita uma engenharia reversa para aproximar o comportamento dele.

5.3.2.3 Página de Publicação

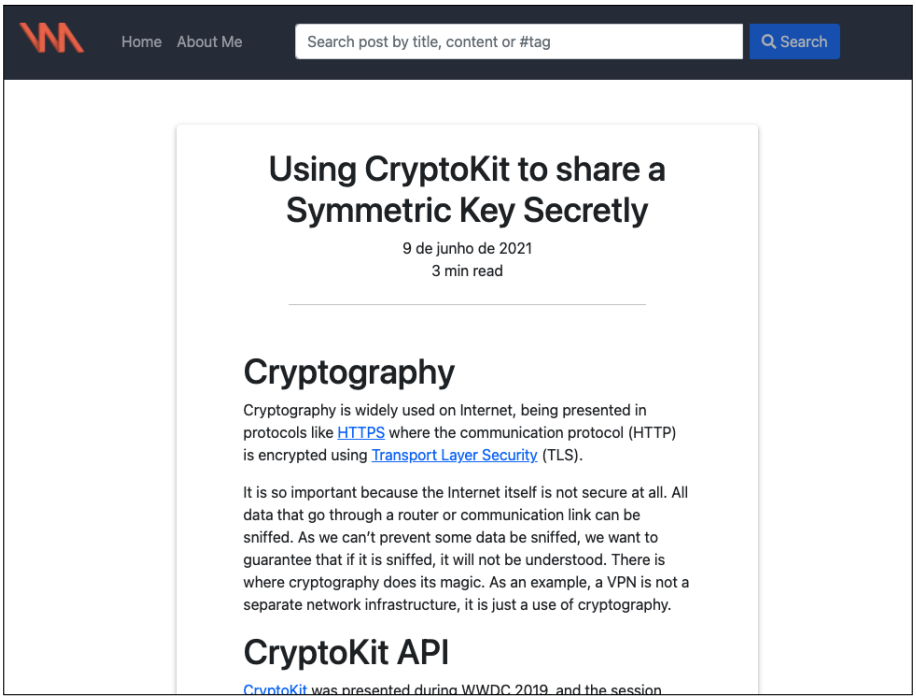
A Figura 5.19 mostra a página de uma publicação. Essa página é gerada dinamicamente no lado do servidor, baseada na requisição do cliente que envia o ID da publicação como parte da URL. Essa página contém todas informações da publicação, além de botões para compartilhamento em redes sociais e um setor de comentários no final dela.

5.3.2.4 Formatação de Texto

Cada publicação é escrita utilizando a linguagem de marcação markdown (Seção 2.24). Tanto o resumo, que é visível na página inicial, quanto o conteúdo completo é armazenado formatado em markdown. É tarefa do cliente fazer o *parse* desta estrutura e apresentar o conteúdo formatado, e para isso foi usada a biblioteca **`marked`** (Seção 2.23). A escolha pelo `marked` se deu por ser uma biblioteca conhecida, bem documentada e leve.

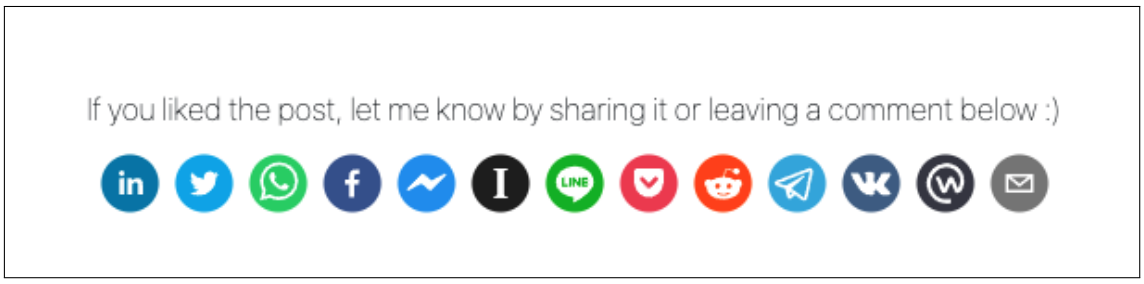
Outra opção para lidar com a formatação seria usar o `marked` na criação da postagem, armazenando ele já formatado em HTML no banco de dados. Porém, a abordagem escolhida é mais escalável para outros tipos de clientes, como mobile, que não utilizam HTML. Desta forma o markdown serve como uma linguagem neutra independente de plataforma.

Figura 5.19: Pagina de publicação



5.3.2.5 Compartilhamento em Redes Sociais

Figura 5.20: Botões de compartilhamento em redes sociais

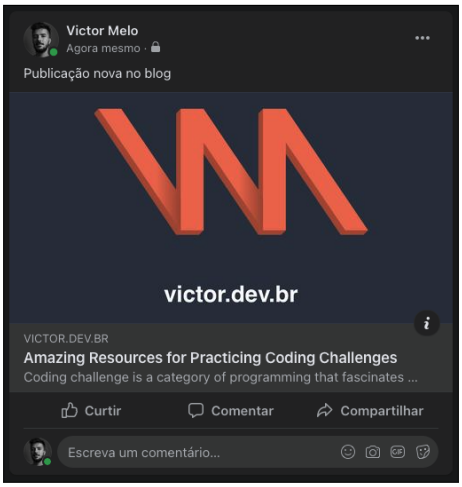


Cada publicação possui um conjunto de botões para compartilhamento em diferentes redes sociais. A Figura 5.20 exibe esses botões. foi usado o *Open Graph Protocol* (Seção 2.35) para que nas redes sociais não seja exibido apenas uma URL, mas sim um conteúdo formatado com título, imagem e descrição (Figura 5.21).

5.3.2.6 Comentários

A funcionalidade de comentários (Figura 5.22) existe no final de cada página de publicação. Para gerenciar os comentários o administrador precisa estar logado, assim

Figura 5.21: Publicação compartilhada no Facebook



exibindo em cada comentário um botão para deletá-lo. Como essa é uma operação destrutiva, ela foi implementada com um alerta de confirmação (Figura 5.23) que aparece antes de efetivamente deletar o comentário da base de dados.

5.3.2.7 Criação, Edição e Deleção de publicação

Existe uma página para criação de novas publicações (Figura 5.24), e ela é acessada através do *dashboard* (Seção 5.3.1.4). Ao preencher o formulário e clicar em confirmar, a publicação se torna visível instantaneamente no blog.

A página de edição é exatamente igual à de criação, sendo a única diferença o fato de que o formulário já começa preenchido com os valores atuais da publicação.

O botão **DELETE** no *dashboard* serve para deletar a publicação. Como essa é uma operação destrutiva, ela também usa um alerta de confirmação (Figura 5.25) antes de efetivamente deletar a publicação da base de dados.

5.3.3 Busca

A funcionalidade de busca inclui tanto a barra superior quanto os menus de tags e datas da *home*. Estes componentes são apresentados a seguir.

Figura 5.22: Seção de comentários

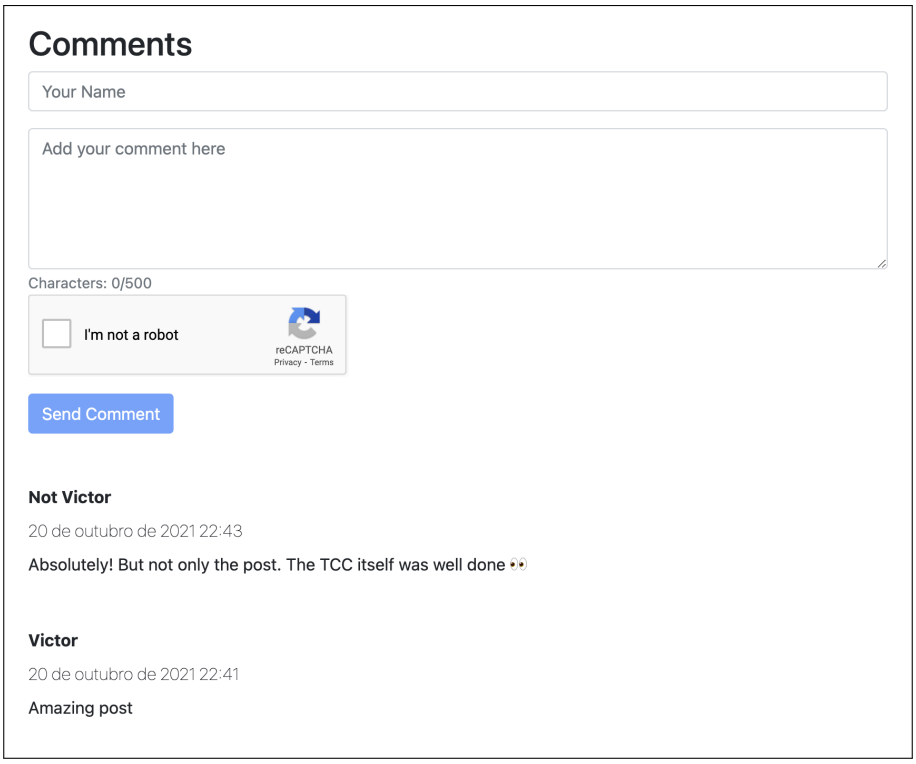


Figura 5.23: Alerta de confirmação de deleção de comentário

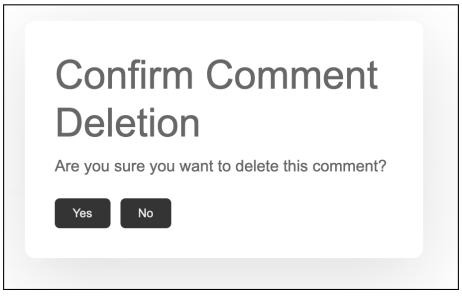


Figura 5.24: Tela de criação de publicação

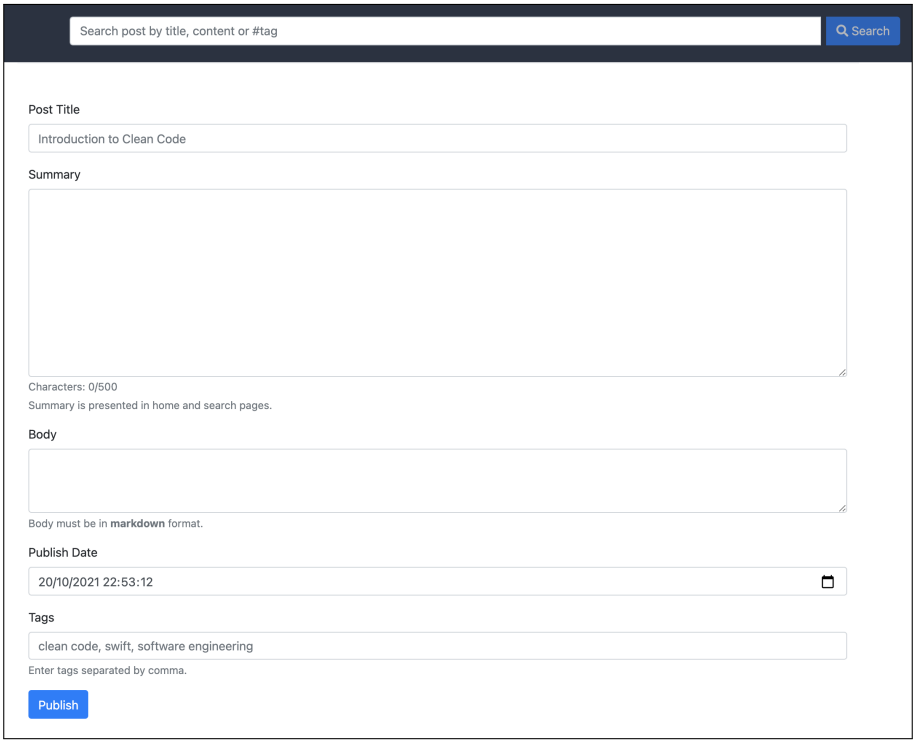
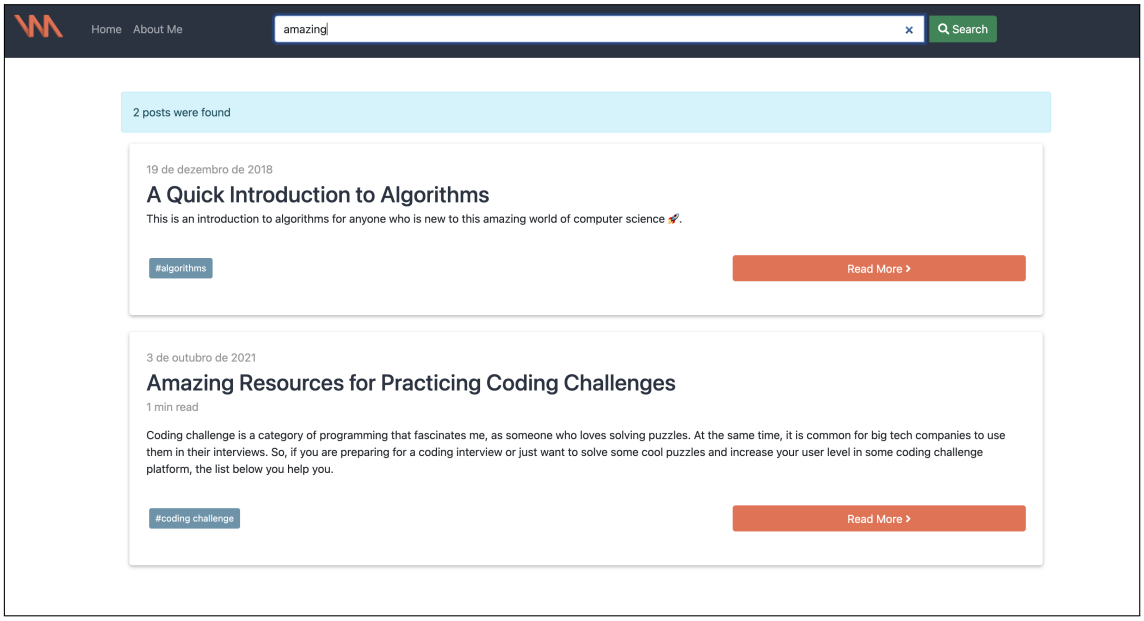


Figura 5.25: Alerta de confirmação de deleção de publicação

5.3.3.1 Barra de Busca

A barra de busca foi implementada para pesquisar em diferentes áreas: no **título**, no **conteúdo**, e nas **tags**. As buscas que começam com cerquilha são interpretadas como buscas por tag, enquanto as restantes pesquisarão tanto no título quanto no conteúdo. Como exemplo, se o usuário pesquisar `#ios`, irá retornar publicações que possuem a tag `ios`. Já se ele pesquisar `ios`, irá buscar por publicações que contém `ios` no título ou no conteúdo. A Figura 5.26 apresenta o resultado da busca pela palavra `amazing`.

Figura 5.26: Tela de resultado de busca

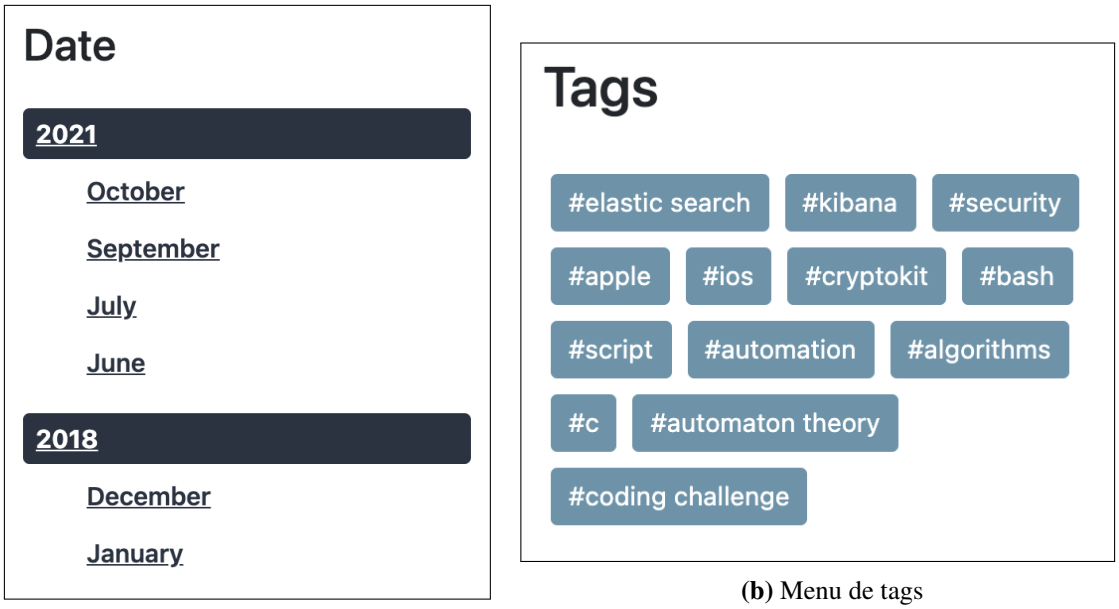


Nessa Figura 5.26 percebe-se um alerta azul dizendo *"2 posts were found"*. Esse alerta é importante para o usuário saber que os dois resultados exibidos são exatamente os retornados. Caso contrário, em situações onde o sistema retorna zero ou um resultado, o usuário poderia pensar que a pesquisa não funcionou.

5.3.3.2 Menu de Datas

O menu de datas (Figura 5.27 (a)) serve para exibir as publicações realizadas no período selecionado. O usuário pode selecionar tanto um mês quanto um ano específico. No servidor o acesso ao banco de dados é feito de forma similar às pesquisas na barra de busca, com apenas um filtro a mais por postagens que possuem `timestamp` dentro do período selecionado. Ao clicar em qualquer período, o usuário é encaminhado para uma tela igual à exibida na Figura 5.26, contendo apenas as postagens do período selecionado.

Figura 5.27: Menus de datas e tags



(a) Menu de datas

(b) Menu de tags

5.3.3.3 Menu de Tags

O menu de tags (Figura 5.27 (b)) exibe todas as tags aplicadas no blog. Futuramente, por escalabilidade, ele será revisitado para exibir apenas as tags mais usadas. Ao selecionar uma tag, o usuário é encaminhado para a página da Figura 5.26, e o resultado é equivalente à buscar a tag na barra de busca.

5.3.4 Sobre mim

A página *about me* (Figura 5.28) tem o propósito de aproximar o autor dos leitores. Ela contém uma descrição, foto e links para as principais redes sociais do autor. Quase todos dados exibidos nela são armazenados no banco de dados MongoDB (Seção 5.7.5), exceto a imagem de perfil que é armazenada num bucket S3 (Seção 5.7.2).

A Figura 5.29 mostra tela de edição dos dados da página *about me*. Ela é acessada pelo administrador através de um botão no *dashboard* (Figura 5.15).

Figura 5.28: Página about me

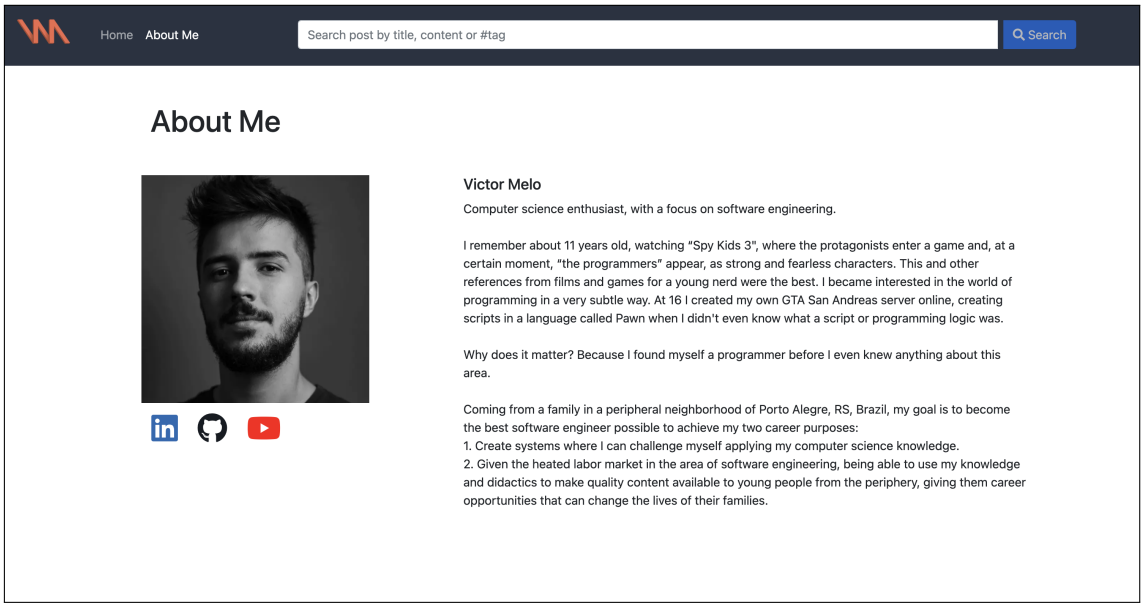
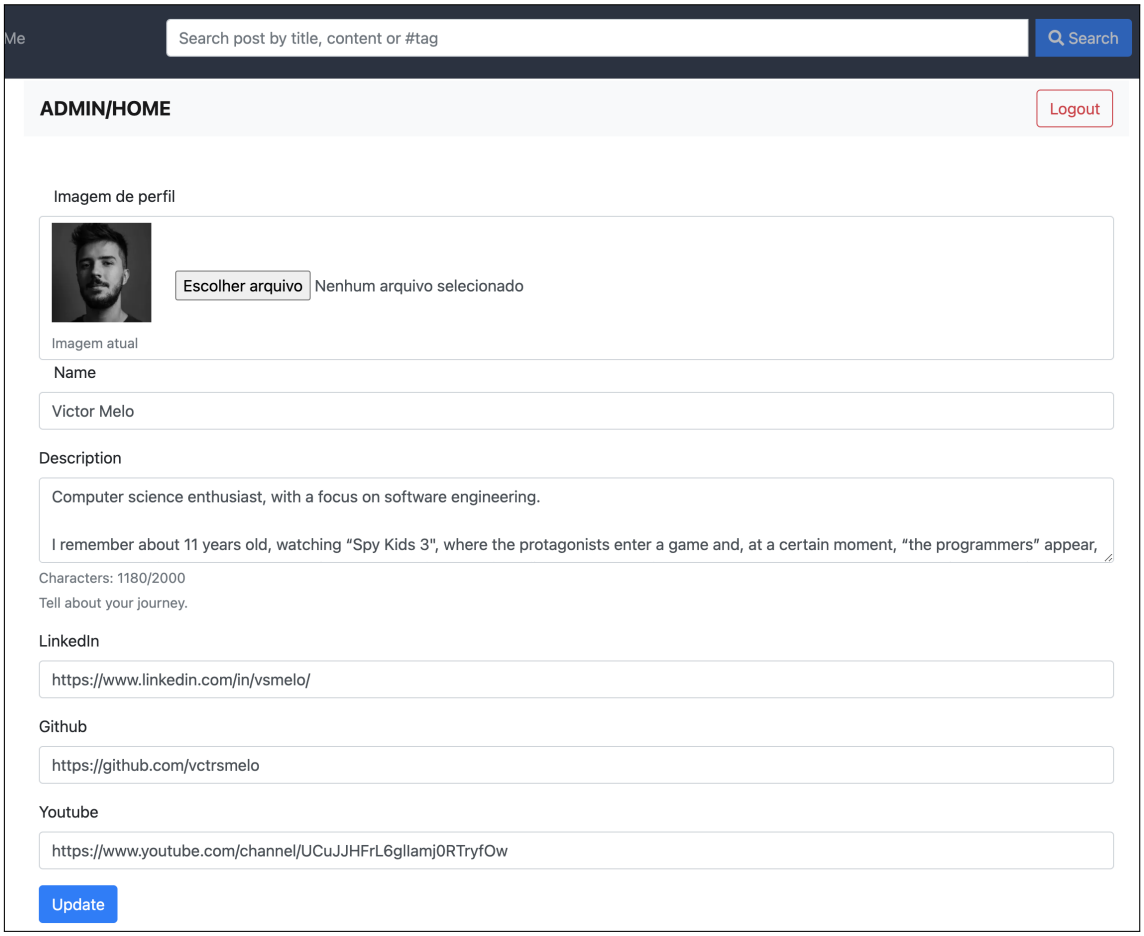


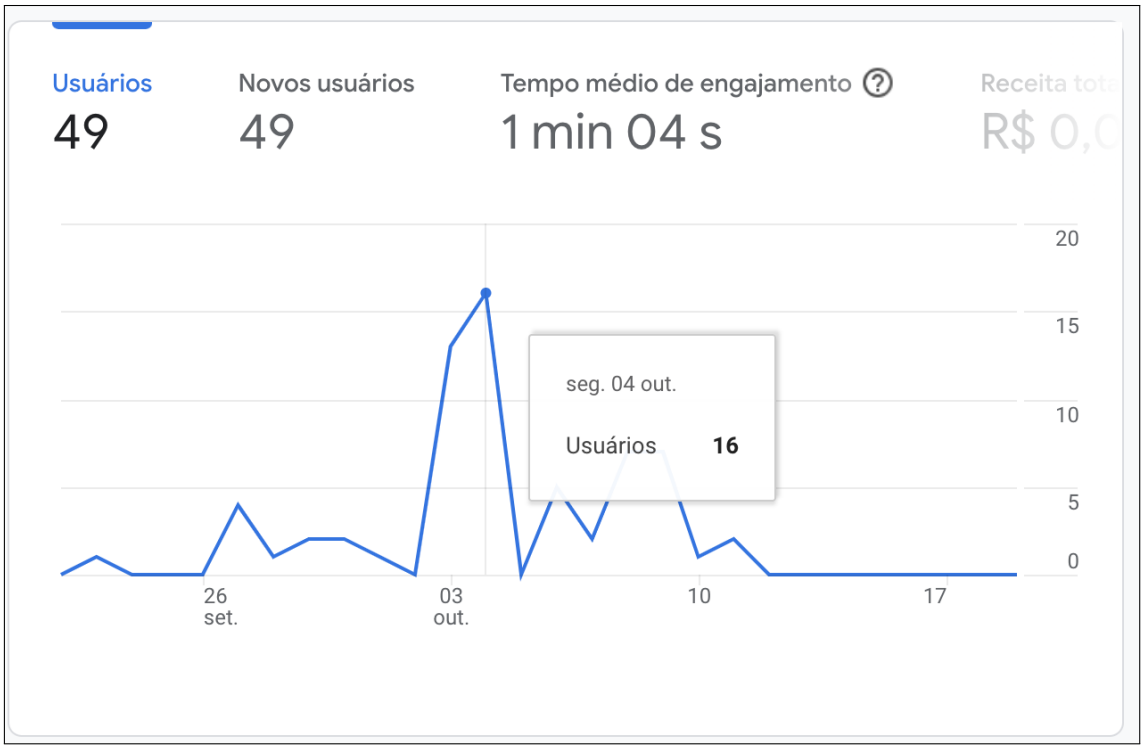
Figura 5.29: Página de edição do about me



5.3.5 Analytics

O *Google Analytics* foi usado como ferramenta de análise. Com ele é possível entender quais publicações ou tags são mais acessadas, de onde partem esses acessos, entre outros dados que podem ajudar a entender melhor o público do blog. Como exemplo de resultados já obtidos, a Figura 5.30 exibe a quantidade de acessos no início de outubro de 2021, enquanto a Figura 5.31 exibe a lista de países e a acessos a partir deles no mesmo período.

Figura 5.30: Visualização de acessos no analytics



5.4 Visão Geral dos Repositórios de Código

Para este projeto, foi utilizado o Github (Seção 2.19). Inicialmente havia um repositório único com todo o código cliente e servidor. Posteriormente ele foi divididos em dois: `vsm.dev-frontend` pro cliente e `vsm.dev-backend` pro servidor. Essa divisão surgiu para facilitar alterações em um sem afetar o outro. O repositório `vsm.dev-frontend` é analisado na seção 5.5, enquanto o `vsm.dev-backend` é analisado na Seção 5.6.

Figura 5.31: Visualização da origem dos acessos no analytics

	País	↓ Usuários	Novos usuários	Sessões engajadas	Taxa de engajamento	Sessões engajadas por usuário	Tempo médio de engajamento	Contagem de eventos Todos os eventos	Conversões Todos os eventos	Receita total
	Totais	49 100% do total	49 100% do total	47 100% do total	55,95% Média de 0%	0,96 Média de 0%	1 min 04 s Média de 0%	980 100% do total	0	R$ 0,00
1	Brazil	35	35	43	61,43%	1,23	1 min 20 s	909	0	R$ 0,00
2	United States	5	5	1	20%	0,20	0 min 04 s	23	0	R$ 0,00
3	China	4	4	0	0%	0,00	0 min 00 s	12	0	R$ 0,00
4	Argentina	1	1	1	100%	1,00	0 min 48 s	5	0	R$ 0,00
5	Canada	1	1	1	100%	1,00	0 min 01 s	6	0	R$ 0,00
6	Germany	1	1	0	0%	0,00	0 min 00 s	3	0	R$ 0,00
7	Ireland	1	1	1	100%	1,00	3 min 51 s	18	0	R$ 0,00
8	Portugal	1	1	0	0%	0,00	0 min 00 s	4	0	R$ 0,00

5.5 Repositório Frontend

Este repositório contém o código que irá gerar a página web exibida ao usuário. Para implementação foi utilizado o Next.js (Seção 2.30) com JavaScript. A seguir esse repositório é apresentado a partir de três perspectivas: a de **estrutura de pastas**, a de **configurações do ambiente** e a de **dependências**.

5.5.1 Estrutura de Pastas

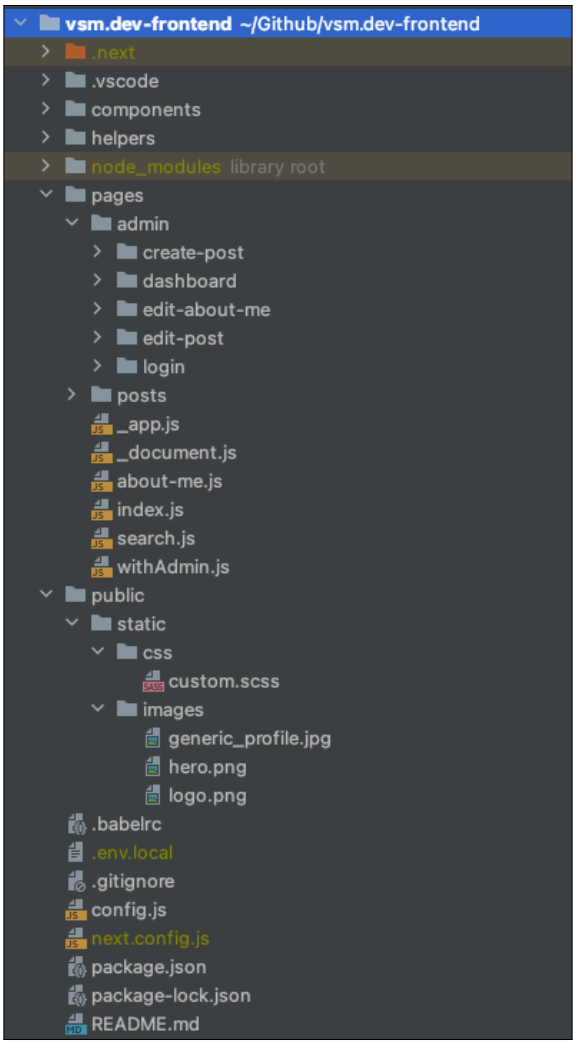
A estrutura de pastas (Figura 5.32) segue o padrão do Next.js. Nela há uma pasta separada chamada **`components`** que mantém o código de elementos visuais reutilizáveis (e.g., barra superior). Já a pasta **`helpers`** contém funções auxiliares, incluindo a lógica para lidar com *cookies*. A pasta **`pages`** contém as páginas efetivamente, e as subpastas dela são: **`admin`**, contendo todas as páginas de administração; e **`posts`** contendo apenas a página de publicação. Já os arquivos dentro de `pages` são páginas diversas e auxiliares, sendo `index.js` a *home*. A pasta **`public`** contém dados estáticos, como arquivos CSS e imagens.

As outras pastas e arquivos são de configuração ou do git.

5.5.2 Configuração do Frontend

O arquivo de configuração `next.config.js` possui as seguintes variáveis:

Figura 5.32: Estrutura de pastas do cliente



- **APP_NAME:** nome do sistema;
- **API:** URL da API executada no *web server* (Seção 5.6);
- **PRODUCTION:** valor booleano indicando se o sistema está executando em ambiente de produção;
- **DOMAIN:** domínio do blog.

Já o arquivo de configuração `.env.local` possui as seguintes variáveis:

- **NEXT_PUBLIC_GOOGLE_ANALYTICS:** chave pública enviada ao serviço de analytics do google;
- **RECAPTCHA_SITE_KEY:** chave pública enviada ao serviço de reCAPTCHA do Google.

5.5.3 Dependências

O projeto contém 15 dependências diretas. Como cada uma pode ter suas próprias dependências, e assim recursivamente, a estrutura de dependências forma um grafo, fazendo com que essas 15 dependências importem no projeto um total de outras 293 dependências. No Github é possível obter informações importantes sobre as dependências, incluindo possíveis vulnerabilidades.

A Tabela 5.1 apresenta cada dependência do front-end com a quantidade de estrelas no Github (indicador de popularidade), a seção do Capítulo 2 onde ela foi descrita e por fim a justificativa de seu uso.

5.6 Repositório Backend

Esta Seção apresenta a implementação do *web server* implementado no repositório `vsm.dev-backend`. A implementação foi realizada utilizando `Node.js` (Seção 2.32) e `express` (Seção 2.14).

A seguir esse repositório é apresentado a partir de cinco perspectivas: a de **estrutura de pastas**, a dos **serviços**, a das **funções auxiliares** a de **configurações do ambiente** e por fim a de **dependências**.

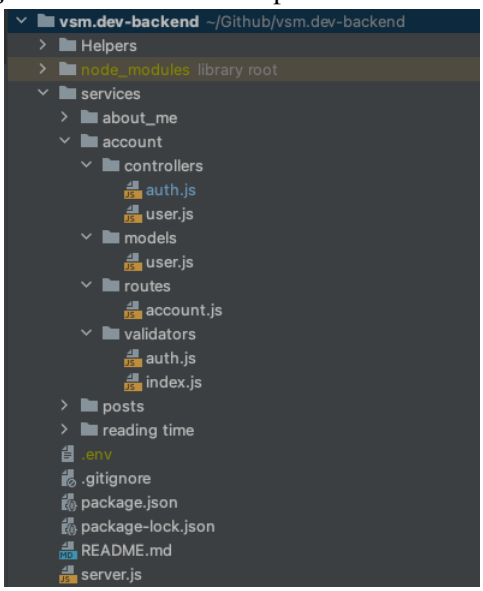
Tabela 5.1: Dependências do código cliente

Dependência	Estrelas	Visto na Seção	Descrição
`axios`	88.6k	2.6	Foi utilizado em todas as comunicação com o servidor.
`styled-components`	35k	2.45	Foi utilizada para estruturar o código de forma mais concisa.
`bootstrap`	154k	2.9	Aplicado em diversos elementos da interface
`js-cookie`	18.7k	2.21	Usado para gerenciar o cookie que mantém o token de login.
`marked`	26.1k	2.23	Converte o markdown já aplicado no conteúdo da publicação em tags HTML.
`next.js`	74.8k	2.30	*Framework* que define toda estrutura do projeto.
`nprogress`	23.5k	2.34	Usado para criar a barra de carregamento superior azul que aparece ao navegar pra outra página.
`prism.js`	9.7k	2.36	É usado para adicionar highlight nos códigos das publicações.
`react`	177k	2.37	Biblioteca de construção de interface.
`react-bootstrap`	20.1k	2.38	Alguns componentes visuais foram definidos a partir de componentes dessa biblioteca.
`react-confirm-alert`	205	2.37	Usado nos alertas de confirmação de deleção de publicação ou comentário.
`react-google-recaptcha`	747	2.40	Usado pra criar o reCAPTCHA dos comentários.
`react-share`	1.9k	2.41	Usada para criar os botões de compartilhar publicação.
`sass`	13.5k	2.43	Usada para criar arquivos de estilo (CSS) com maior

5.6.1 Estrutura de Pastas

A estrutura de pastas (Figura 5.33) contém as seguintes pastas: a **`Helpers`** contém arquivos com funções utilizadas em diferentes partes do sistema, e a Seção 5.6.3 discorre sobre ela. A pasta **`node_modules`** mantém as dependências do projeto. A pasta **`services`** contém os serviços implementados, cada um em sua própria subpasta, e a Seção 5.6.2 discorre sobre esses serviços.

Figura 5.33: Estrutura de pastas do web server.



5.6.2 Serviços

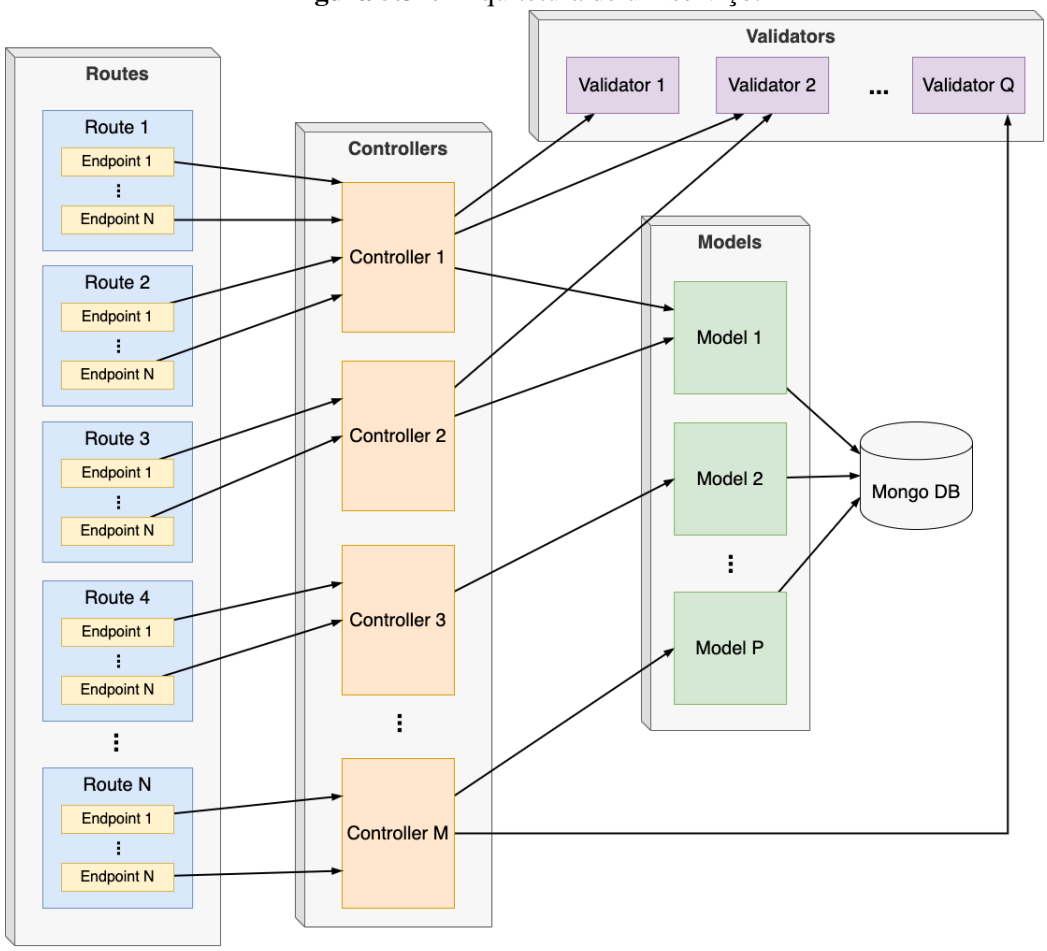
Foram implementados três serviços para prover as funcionalidades do blog, e cada um possui os seguintes elementos:

- **Route:** apresentado na Seção 2.42;
- **Endpoint:** apresentado na Seção 2.42;
- **Controller:** conjunto de funções, onde cada função executa uma operação solicitada pelos *endpoints*;
- **Validator:** é um *middleware* que valida os dados recebidos do cliente;
- **Models:** representam os dados armazenados no banco de dados. Eles são implementados com o Mongoose (Seção 2.27).

A Figura 5.34 apresenta a arquitetura dos serviços implementados no *web server*.

A partir dela conseguimos identificar as seguintes relações entre os componentes:

Figura 5.34: Arquitetura de um serviço.



- Um `endpoint` referencia um e apenas um `controller`. Porém um `controller` pode ser referenciado por mais de um `endpoint`.
- Todos `endpoints` de uma `route` referenciam o mesmo `controller`. Porém um `controller` pode ser referenciado por `endpoints` de diferentes `routes`.
- Um `controller` referencia zero ou mais `validators`.
- Um `controller` referencia um e apenas um `model`. Porém um `model` pode ser referenciado por mais de um `controller`.
- Cada `model` possui uma relação direta com o banco de dados.

Cada serviço respeita essa arquitetura, sendo que alguns não possuem `validators`. Nas subseções abaixo cada serviço é descrito.

Tabela 5.2: Requisições do `about me service`

Requisição	Privilégio	Objetivo
GET `/about_me?email`	aberto	Retorna os dados do perfil da página `About Me`.
POST `/about_me/edit?email`	admin	Edita os dados da página *about_me*.

Tabela 5.3: Requisições do `account service`

Requisição	Privilégio	Objetivo
GET `/user/info` auth:{token}	usuário logado	Retorna dados do usuário logado a partir do token enviado.
GET `/admin` auth:{token}	admin	Retorna dados do administrador logado. Usado para exibir ou não as telas de administração.
POST `/admin/login` body: {email, password}	aberto	Usado para fazer login no sistema.

5.6.2.1 About Me Service

Este serviço tem o propósito de lidar com as informações exibidas na página *about me* do blog. Ele possui duas funcionalidades, e cada uma pode ser invocada através de uma requisição própria. A Tabela 5.2 exibe as requisições.

5.6.2.2 Account Service

Este serviço diz respeito a todas as funcionalidades relacionadas com a conta de usuário. São três requisições disponíveis, vistas na Tabela 5.3.

5.6.2.3 Posts Service

Este é o serviço que contém a maior quantidade de requisições. Nele estão contidas: as requisições relativas às publicações do blog (Tabela 5.4); as requisições relativas aos comentários (Tabela 5.5); e as requisições relativas à administração das publicações (Tabela 5.6).

5.6.3 Funções Auxiliares

A pasta **`Helpers`** contém algumas funções auxiliares: o arquivo `Logger.js` contém a classe que encapsula a ferramenta de log usada; o arquivo `reCaptcha.js`

Tabela 5.4: Requisições de publicações do `post service`

Requisição	Privilégio	Objetivo
GET `/posts/id/:postId`	aberto	Retorna publicação que possui o id `postId`.
GET `/posts?page&size`	aberto	Retorna publicações da página `page`, considerando que cada página contém `size` publicações.
GET `/posts/tags`	aberto	Retorna todas as tags existentes.
GET `/posts/dates`	aberto	Retorna todas as datas das publicações existentes.

Tabela 5.5: Requisições de comentários do `post service`

Requisição	Privilégio	Objetivo
GET `/:postId/comments`	aberto	Retorna lista de comentários da publicação com id `postId`.
POST `/:postId/comments/add`	aberto	Envia um comentário novo na publicação com id `postId`.
DELETE `/:postId/comments/delete` body: {commentId}	admin	Deleta o comentário com id `commentId` da publicação com id `postId`.

Tabela 5.6: Requisições de administração de publicações do `post service`

Requisição	Privilégio	Objetivo
POST `/posts/edit` auth:{token} body:{tags, title, summary, body, date, id}	admin	Edita a publicação que possui o `id` recebido no body.
POST `/posts/create` auth:{token} body:{tags, title, summary, body, date}	admin	Cria uma nova publicação.
DELETE `/posts/delete?id` auth:{token}	admin	deleta a publicação que possui o `id` recebido como parâmetro.

possui a função que chama o serviço do Google para validar se um token recebido é válido; o último arquivo `strings_validators.js` possui funções para validar strings, e atualmente contém apenas validações para ver se uma string é `undefined`, `null` ou vazia.

5.6.4 Configuração do Web Server

O projeto possui um arquivo `.env` que contém as seguintes variáveis de configuração:

- **PORT:** porta onde o *web server* será executado;
- **CLIENT_URL:** URL do cliente que poderá se comunicar com o *web server*;
- **DATABASE_CLOUD:** URI (Seção 2.46) para se conectar com o banco de dados;
- **AWS_ACCESS_KEY_ID:** chave de acesso pública aos recursos da AWS;
- **AWS_SECRET_ACCESS_KEY_ID:** chave de acesso secreta aos recursos da AWS;
- **AWS_REGION:** região da AWS;
- **EMAIL_ADMIN_TO:** e-mail que irá receber notificação de tentativa de login;
- **EMAIL_ADMIN_FROM:** e-mail que irá enviar notificação de tentativa de login;
- **JWT_SECRET:** segredo para gerar token JWT usado no login (Seção 5.3.1.1);
- **RECAPTCHA_SECRET_KEY:** chave secreta do blog enviada ao serviço de re-CAPTCHA do Google;

5.6.5 Dependências

São 14 dependências diretas, com 206 indiretas. A Tabela 5.7 exibe a lista de dependências diretas.

5.7 Infraestrutura

Para infraestrutura do sistema, foi escolhida a Amazon Web Services (Seção 2.1). A AWS é amplamente usada por diversas empresas, como Netflix, Autodesk, Zoom, Samsung, Disney, entre outras gigantes (AMAZON, 2021a; AMAZON, 2021b; AMAZON,

Tabela 5.7: Dependências do código servidor

Dependência	Estrelas	Visto na Seção	Descrição
`aws-sdk`	6.8k	2.5	Aplicado na comunicação com os serviços AWS usados.
`axios`	88.6k	2.6	Usado nas comunicações RESTful com cliente.
`express-cors`	5.1k	2.15	Usado como *middleware* de segurança na comunicação com cliente. Garante autenticidade do cliente.
`dotenv`	14.3k	2.12	Facilita o gerenciamento das variáveis de ambiente.
`express-jwt`	4.1k	2.16	Usado como *middleware* para validar o token JWT de sessão recebido do cliente.
`express-validator`	5.1k	2.17	Usado para validar se e-mail e senha recebidos no login estão formatados de forma válida.
`formidable`	6k	2.18	Usado na funcionalidade de edição do `about me`, para parsear os dados recebidos do cliente.
`mongoose`	23.4k	2.27	Usado para comunicação com banco de dados MongoDB.
`mongoose-paginate-v2`	358	2.28	Usado para buscar do banco de dados as publicações já paginadas pra página inicial.
`morgan`	6.7k	2.29	Ferramenta de log usada para requisições HTTP.
`nodemon`	23.2k	2.33	Usado pra facilitar o desenvolvimento

2021c; AMAZON, 2021d). Por conta dessa demanda, a escolha da AWS se deu pelo interesse pessoal no estudo dessa ferramenta, a fim de obter conhecimento extremamente útil para além deste projeto.

Dentre os diversos serviços providos pela AWS, três foram utilizados: *Elastic Computing (EC2)* (Seção 2.2); *Simple Cloud Storage (S3)* (Seção 2.4); e por último *Simple Email Service (SES)* (Seção 2.3).

Para usar os serviços da AWS é necessário escolher uma região. Nesse projeto foi escolhida *North Virginia* (us-east-1). A escolha foi feita considerando dois fatores: **alta disponibilidade** e **preço**.

5.7.1 Uso do Elastic Computing (EC2)

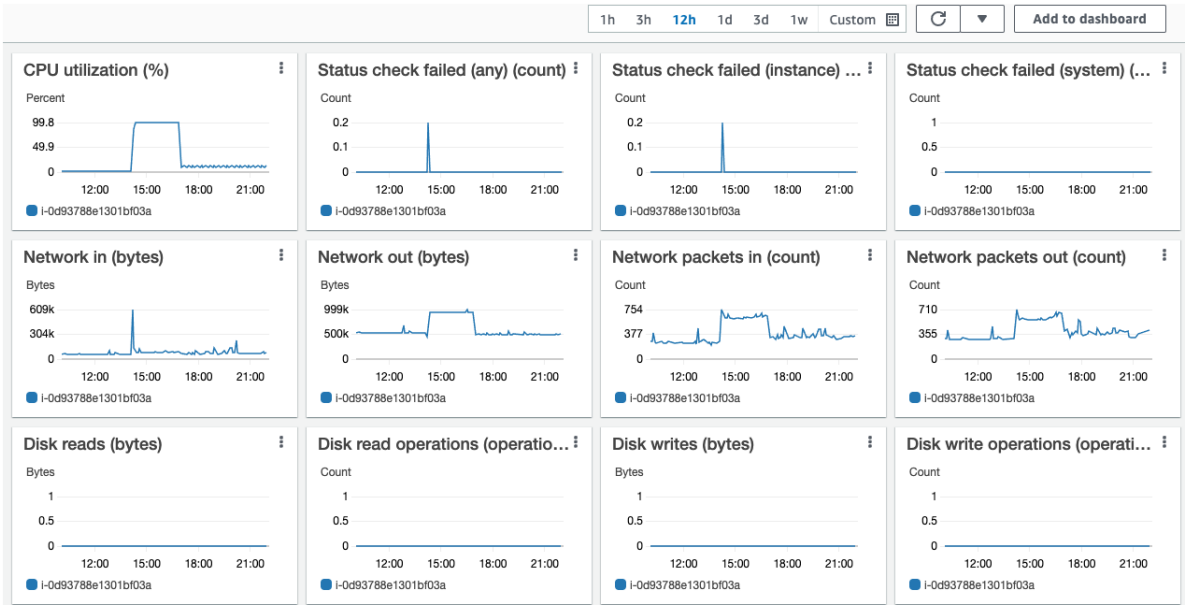
O EC2 foi utilizado para subir uma máquina virtual na nuvem. Essa máquina provê o *web server*, e possui as seguintes características:

- **Instance type:** t2.micro
- **Platform:** Ubuntu
- **IP Público:** 34.204.40.74
- **Armazenamento:** 8 GB.

Foram abertas três portas para acesso direto a partir de qualquer IP. As portas abertas foram a 22, 80 e 443. A porta 22 é usada para acesso seguro via SSH. É por ela que é possível acessar a máquina remotamente via terminal e executar comandos para subir os serviços. Já a porta 80 é do HTTP e, apesar de estar aberta, o site não é acessado via HTTP. Internamente é feito um redirecionamento para a porta 443, que recebe requisições HTTPS.

No console da AWS é possível monitorar o estado da máquina virtual. Na Figura 5.35 são exibidas algumas métricas que podem ser acompanhadas, onde percebemos um aumento do uso dos recursos no período entre 14h e 17h. Durante esse período o blog estava sendo alterado, o que justifica esse aumento temporário.

Figura 5.35: Monitoramento de um período de 12 horas.



5.7.2 Uso do Simple Cloud Storage (S3)

O S3 foi escolhido como banco de dados para armazenamento de objetos não estruturados. Por enquanto, o S3 foi utilizado para armazenar duas imagens: a imagem do perfil exibida na página *about me* (Seção 5.3.4); e a imagem usada no compartilhamento das publicações em redes sociais (Seção 5.3.2.5).

5.7.3 Uso do Simple Email Service (SES)

O SES foi utilizado para disparar um e-mail sempre que alguém tenta fazer login como administrador. No projeto o serviço está sendo usado em modo *sandbox*, o que significa que há um limite máximo de 200 e-mails por dia e um limite de transmissão de 1 e-mail por segundo. Estas restrições não devem afetar o sistema, dado que é esperado um número muito menor de e-mails enviados por dia.

5.7.4 Proxy Reverso com Nginx

Para que a instância EC2 funcione como um *web server*, foi usado o Nginx (Seção 2.31) como proxy reverso. Essa abordagem é interessante pois permite trocar o endereço IP ou a porta do *web server* de forma mais fácil.

A configuração do Nginx está conforme a Figura 5.36. Nela podemos identificar dois servidores distintos. O primeiro escuta a porta 443 (HTTPS), e ao receber requisições faz dois tipos de encaminhamento: se não tiver sido passado nenhum caminho na URL, irá encaminhar o front-end na porta 3000; se tiver recebido o caminho `/api/` irá encaminhar para o serviço do *web server* na porta 8000.

O segundo servidor existe apenas para fazer um redirecionamento, indicando que o site está disponível apenas via HTTPS. Isso serve para redirecionar o usuário automaticamente caso ele tente acessar o site via HTTP puro.

Figura 5.36: Configuração do Nginx.

```
server {
  listen 443 ssl default_server;
  server_name victor.dev.br www.victor.dev.br;

  # react app & front-end files
  location / {
    proxy_pass http://localhost:3000/;
  }

  # node api reverse proxy
  location /api/ {
    proxy_pass http://localhost:8000/api/;
  }


#    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/victor.dev.br/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/victor.dev.br/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}

server {
    if ($host = www.victor.dev.br) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    if ($host = victor.dev.br) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    server_name victor.dev.br www.victor.dev.br;
    listen 80;
    return 404; # managed by Certbot
}
```

5.7.5 Banco de dados com MongoDB

O MongoDB (Seção 2.25) foi utilizado como banco de dados principal para armazenar os dados do blog. Dois critérios principais impactaram na escolha por um banco de dados não relacional:

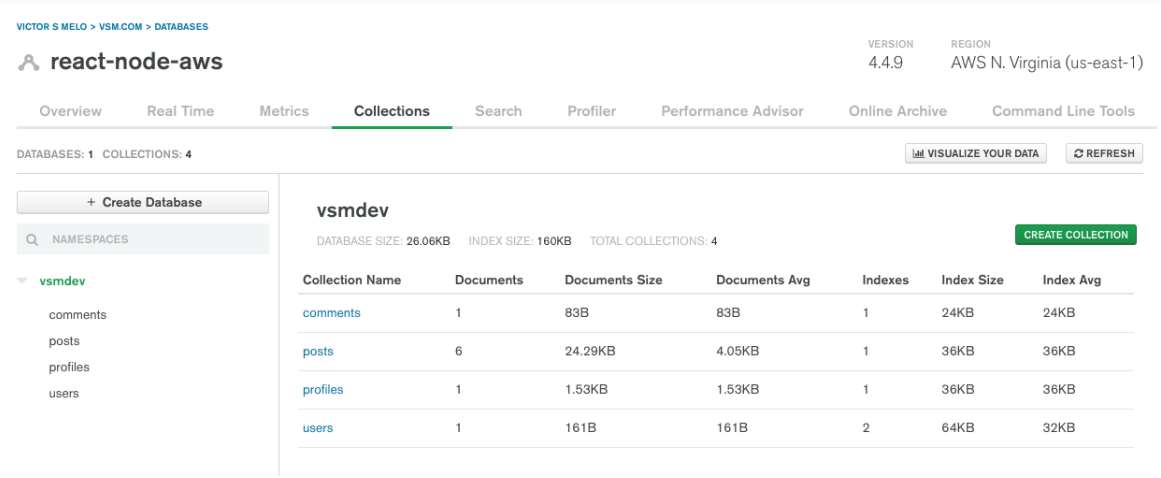
- **Flexibilidade**: com um banco não relacional é mais fácil modificar o esquema;
- **Simplicidade**: o blog, como foi planejado, não possui relações suficientes que justifiquem o uso de um banco relacional.

Como o projeto nasce pequeno e há muita incerteza sobre como irá evoluir, ele se beneficia muito do uso de um banco não relacional.

Dentre os bancos de dados não relacionais, o MongoDB foi escolhido pois a forma de fazer *queries* parecia estar mais próxima de como se faz usando SQL, tendo uma curva de aprendizado menor para quem está acostumado com SQL.

O banco foi implementado em nuvem utilizando o **MongoDB Atlas** (Seção 2.26). O Atlas hospeda o banco na AWS. A Figura 5.37 exibe o painel de controle do Atlas.

Figura 5.37: Visualização do painel de controle no Atlas.



O esquema implementado pode ser visto na Figura 5.38. As subseções abaixo descrevem cada entidade, chamada de *collection* no MongoDB.

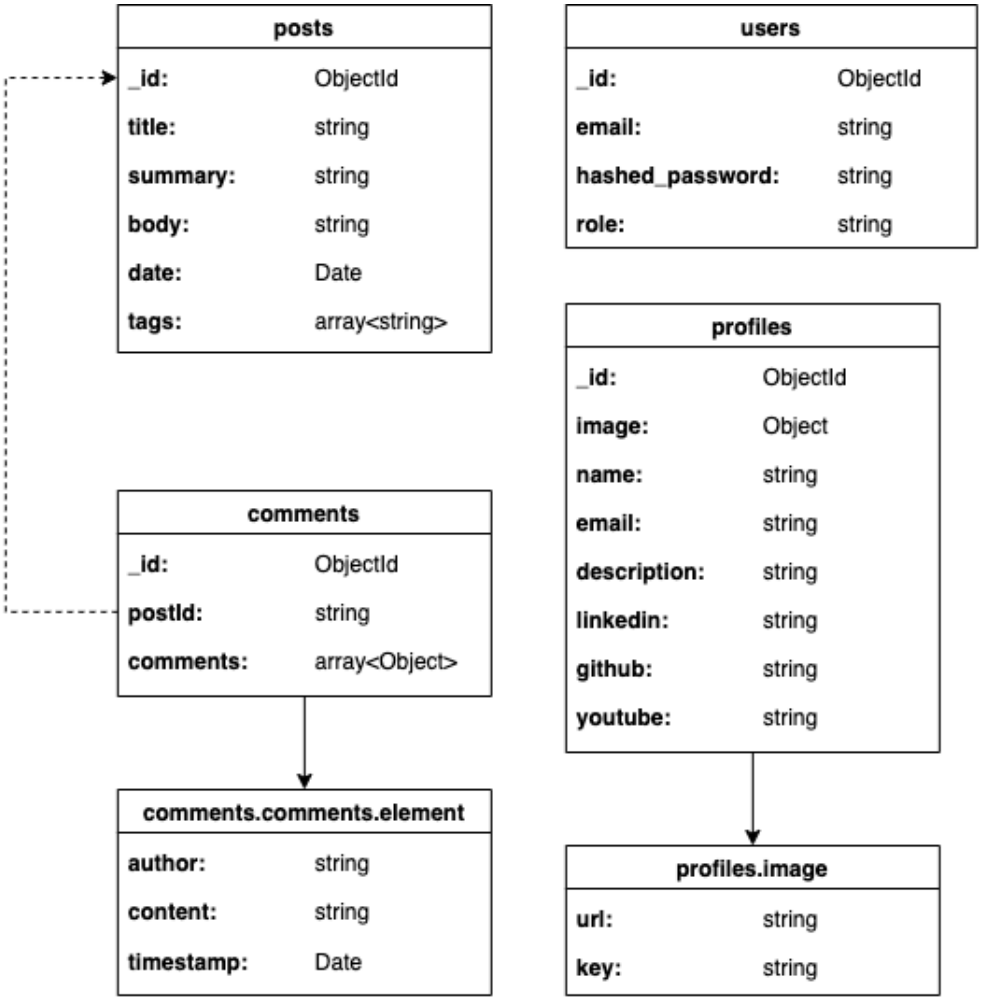
5.7.5.1 posts

A *collection* **posts** armazena todas as publicações do blog. Abaixo segue a descrição de cada atributo nela:

- **_id:** identificador único definido pelo MongoDB;
- **title:** título da publicação;
- **summary:** resumo do conteúdo da publicação;
- **body:** conteúdo completo da publicação;
- **date:** data de criação da publicação;
- **tags:** lista de tags da publicação.

O `summary` é utilizado na página inicial do blog, a fim de despertar o interesse do leitor em ler o conteúdo completo. Tanto o `summary` quanto o `body` são armazenados em markdown (Seção 2.24). O tipo Date do atributo `date` é nativo do MongoDB, e

Figura 5.38: Esquema de dados implementado no MongoDB.



funciona de forma similar à outros tipos equivalentes em outros sistemas. As **`tags`** são armazenadas como `string`, e pela característica não relacional, a mesma tag pode estar repetida em diferentes collections no banco.

5.7.5.2 comments

A *collection* **comments** armazena todos comentários feitos no blog. Cada entrada nela contém a lista de comentários para um publicação específica. Abaixo segue a descrição de cada atributo:

- **_id:** identificador único definido pelo MongoDB;
- **postId:** id do publicação que contém os comentários do atributo **comments**;
- **comments:** lista de comentários para o publicação referenciada pelo postId.

Cada elemento dentro do atributo `comments` é um objeto também, e ele possui os seguintes atributos:

- **author:** nome do autor do comentário;
- **content:** conteúdo do comentário;
- **timestamp:** momento exato do comentário.

5.7.5.3 users

A *collection* **users** é utilizada atualmente para manter apenas a conta do administrador do blog. Os atributos são descritos abaixo:

- **_id:** identificador único definido pelo MongoDB;
- **email:** e-mail utilizado para login;
- **hashed_password:** mantém a senha cifrada, e é utilizada apenas para login. Ela é cifrada com SHA-256.
- **role:** indica o nível de permissão do usuário (e.g., admin).

5.7.5.4 profiles

A *collection* **profiles** mantém o perfil de usuário, e está relacionada com os usuários existentes na *collection* users. Atualmente ela mantém apenas as informações da página *about me*. Seus atributos são:

- **_id:** identificador único definido pelo MongoDB;

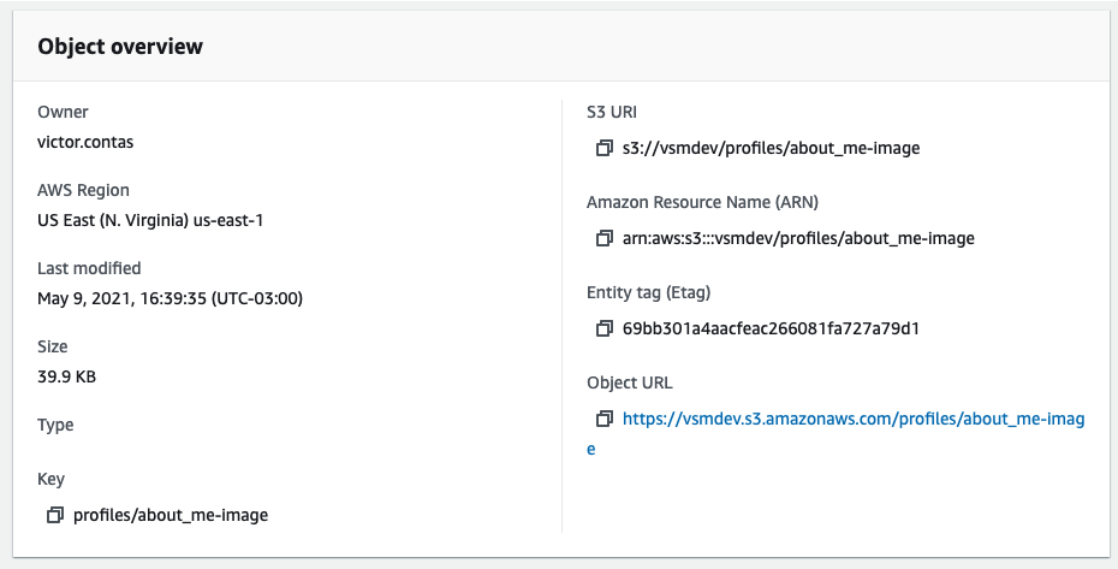
- **image:** foto do perfil;
- **name:** nome do perfil;
- **email:** e-mail do perfil;
- **description:** descrição do perfil;
- **linkedin:** url do LinkedIn do perfil;
- **github:** url do Github do perfil;
- **youtube:** url do canal no Youtube do perfil.

O atributo `image` é um objeto que possui os atributos abaixo:

- **url:** url da imagem no bucket S3;
- **key:** identificador único da imagem no bucket S3.

A Figura 5.39 mostra como essa imagem é armazenada no S3.

Figura 5.39: Imagem do perfil no bucket S3.



5.8 Automações

Seguindo a cultura DevOps (Seção 2.11), foram utilizados alguns *scripts* em `Bash` para facilitar o desenvolvimento, configuração e *deploy* do blog. São quatro *scripts* no total, descritos nas subseções abaixo.

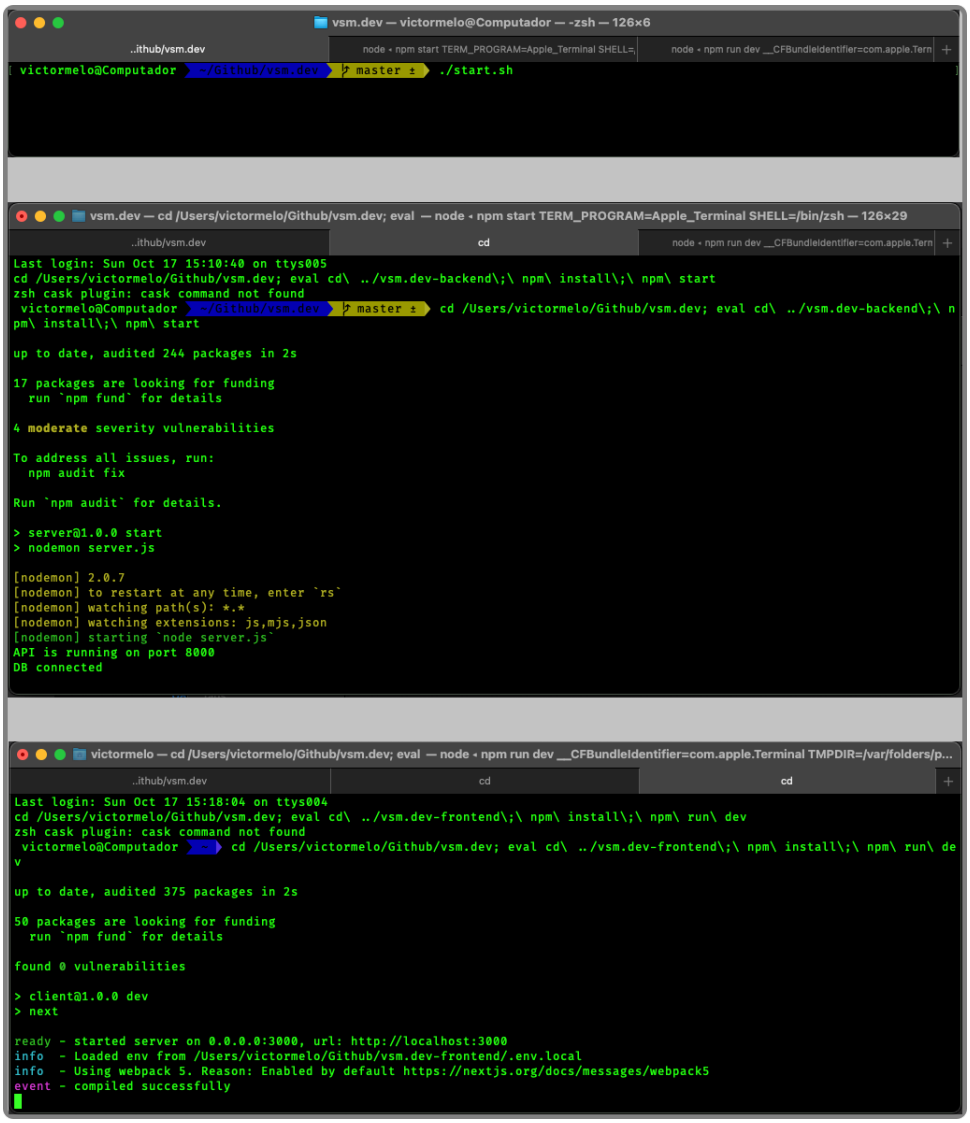
5.8.1 Script start.sh

Este *script* foi usado durante o desenvolvimento. Seu propósito é iniciar o `front-end` e o `back-end` localmente, em abas separadas do terminal. O código implementado segue abaixo:

```
1 ttab -t 'Client' 'cd ../vsm.dev-frontend; npm install; npm run
  ↪  dev' &
2 ttab  -t 'Server' 'cd ../vsm.dev-backend; npm install; npm
  ↪  start'
```

A Figura 5.40 mostra o resultado em cada aba do terminal. A primeira aba é quem executa o `start.sh`, enquanto a segunda roda o `vsm.dev-backend` e a terceira o `vsm.dev-frontend`.

Figura 5.40: Resultado em cada aba ao rodar o *script* start.sh



5.8.2 Script connect_ec2.sh

Este *script* serve para acessar a máquina EC2 remotamente. Ele simplesmente executa um comando `ssh`, passando a chave secreta como parâmetro e o endereço da máquina. O *script* segue abaixo:

```
1 ssh -i ../secrets/ec2-amazon-linux.pem
  ↪ ubuntu@ec2-34-204-40-74.compute-1.amazonaws.com
```

5.8.3 Script configure_ec2.sh

Este *script* foi encontrado na internet, e é o maior dentre todos os quatro. Ele instala e configura todas ferramentas necessárias para o *web server*, incluindo o `Node.js`, o `MongoDB`, o `Nginx` e o `PM2`, apresentados no capítulo 2. Como é um *script* de terceiro, não o exibirei aqui. Caso haja curiosidade, ele pode ser visto em (SETUP...,).

5.8.4 Script deploy.sh

O último script aqui citado foi definido para automatizar o processo de deploy de uma nova versão do blog. Ele é executado dentro da máquina virtual EC2. Ele para todos processos que mantém o site no ar, puxa as versões atualizadas dos repositórios no Github, para por fim reiniciar os processos com o código atualizado. A implementação dele pode ser vista no Anexo B.

6 AVALIAÇÃO

Para avaliação do projeto foi realizado um teste de usabilidade através de um formulário online. Este formulário se dividiu em seis seções: uma para identificar o perfil da pessoa, quatro para realização de atividades, e uma última para obter percepções gerais sobre o blog.

O formulário foi compartilhado em redes sociais pessoais e em um grupo interno da Thoughtworks, sendo que a maior parte das respostas vieram deste último grupo. Foram obtidas 146 respostas, e nas seções a seguir analisamos os dados obtidos.

6.1 Perfil

Nessa seção foram feitas seis perguntas, exibidas abaixo.

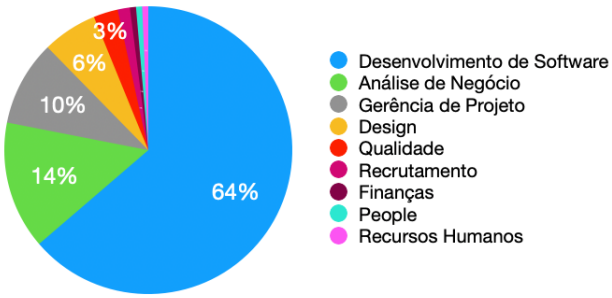
6.1.1 Com qual área você mais se identifica?

Essa pergunta visa entender os interesses e especialização de quem preencheu o formulário. O resultado está na Figura 6.1, onde percebemos que 64% são pessoas desenvolvedoras de software, 14% são analistas de negócio, 10% são gestoras de projeto e os outros 12% se distribuem entre outras áreas.

Figura 6.1: Distribuição por área de atuação

Com qual área você mais se identifica?

Opções	Total	%
Desenvolvimento de Software	93	64%
Análise de Negócio	21	14%
Gerência de Projeto	14	10%
Design	9	6%
Qualidade	4	3%
Recrutamento	2	1%
Finanças	1	1%
People	1	1%
Recursos Humanos	1	1%
	146	**100%**



6.1.2 Com qual gênero você mais se identifica? (opcional)

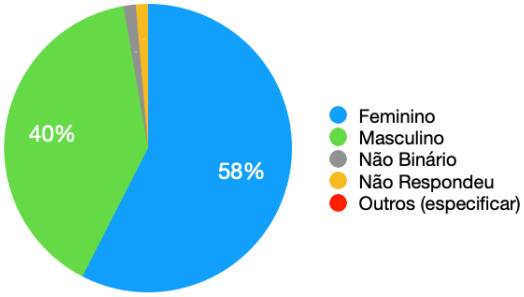
Apesar dessa ser uma pergunta relativamente comum para mapear perfil, ela foi marcada como opcional por duas razões: primeiro por considerar que algumas pessoas podem se sentir desconfortáveis com ela; segundo porque as perguntas deste teste não são variáveis dependentes de gênero.

A Figura 6.2 apresenta o resultado dessa pergunta, onde 144 pessoas responderam. Vemos que 58% das pessoas se identificam como mulheres, 40% como homens e 1% como pessoas não-binárias.

Figura 6.2: Distribuição por gênero

Com qual gênero você mais se identifica?

Opções	Total	%
Feminino	84	58%
Masculino	58	40%
Não Binário	2	1%
Não Respondeu	2	1%
Outros (especificar)	0	0%
	146	100%



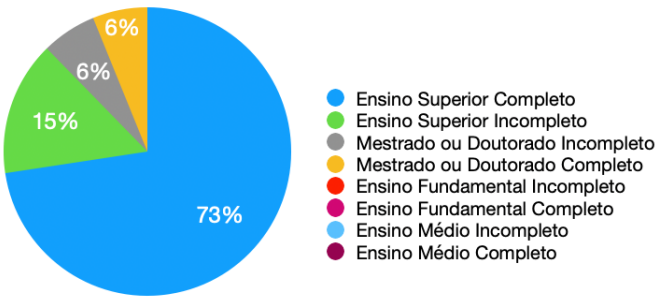
6.1.3 Nível de escolaridade

Essa pergunta visa mapear o nível de escolaridade das pessoas. Na Figura 6.3 vemos que 73% possuem ensino superior completo, 15% ensino superior incompleto, 6% mestrado ou doutorado incompleto e outros 6% mestrado ou doutorado completo.

Figura 6.3: Distribuição por escolaridade

Nível de escolaridade

Opções	Total	%
Ensino Superior Completo	106	73%
Ensino Superior Incompleto	22	15%
Mestrado ou Doutorado Incompleto	9	6%
Mestrado ou Doutorado Completo	9	6%
Ensino Fundamental Incompleto	0	0%
Ensino Fundamental Completo	0	0%
Ensino Médio Incompleto	0	0%
Ensino Médio Completo	0	0%
	146	100%



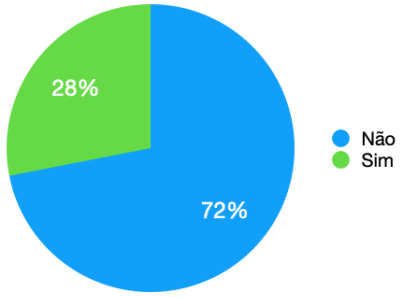
6.1.4 Atualmente você está matriculado em alguma instituição de ensino superior?

Tentando mapear se as pessoas que responderam são estudantes a partir do nível superior, constatamos na Figura 6.4 que 73% delas não estão matriculadas em uma instituição de ensino superior.

Figura 6.4: Distribuição por matrícula em instituição superior

Atualmente você está matriculado em alguma instituição de ensino superior?

Opções	Total	%
Não	105	72%
Sim	41	28%
	146	100%



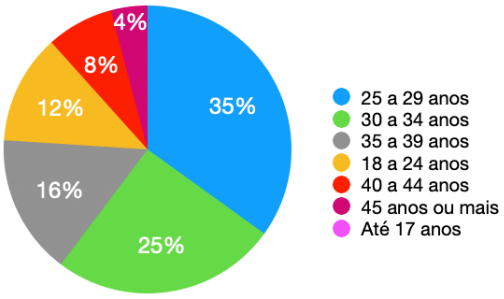
6.1.5 Qual sua faixa etária?

A pergunta da faixa-etária é importante pois pode impactar na forma com que se utiliza as tecnologias. Na Figura 6.5 vemos o resultado, percebendo que cerca de 60% das pessoas estão entre 25 e 34 anos, sendo a maior parte até 29.

Figura 6.5: Distribuição por faixa etária

Qual sua faixa etária?

Opções	Total	%
25 a 29 anos	51	35%
30 a 34 anos	37	25%
35 a 39 anos	23	16%
18 a 24 anos	18	12%
40 a 44 anos	11	8%
45 anos ou mais	6	4%
Até 17 anos	0	0%
	146	100%



6.1.6 Qual equipamento você está usando para responder este formulário?

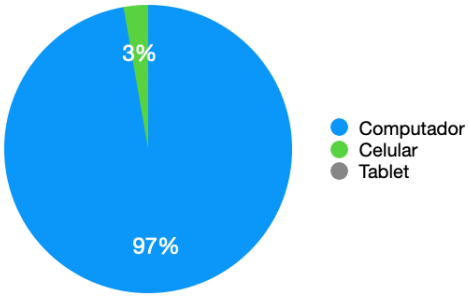
A informação do equipamento usado é importante pra entender se o usuário irá interagir com a versão *mobile* ou *desktop* da interface. Na Figura 6.6 vemos que 97%

usaram computador.

Figura 6.6: Distribuição por equipamento usado

Qual equipamento você está usando para responder este formulário?

Opções	Total	%
Computador	142	97%
Celular	4	3%
Tablet	0	0%
	146	**100%**



6.2 Atividade 1

A primeira atividade era para, a partir da tela inicial da página, navegar até a página da publicação entitulada "A Practical Introduction to Elasticsearch with Kibana". A hipótese a ser validada era a de que as pessoas usariam a barra superior de busca para encontrar essa publicação.

Há diferentes maneiras de completar essa atividade, e a Figura 6.7 mostra a primeira pergunta após a realização da dela. Nessa pergunta queremos entender onde, instintivamente, o usuário tentou ir para chegar na página destino. Percebemos aqui que a grande maioria apenas rolou a tela até encontrar a publicação, enquanto apenas 17% tentou primeiro a barra de busca conforme era esperado, indicando um possível ponto de atenção na análise.

As próximas quatro perguntas foram definidas para serem respondidas usando a **Escala de Likert** (2.13). Essas perguntas e suas respostas podem ser vistas na Figura 6.8.

Foram planejadas duas afirmações positivas e duas negativas, e elas foram definidas para que pudessem ser reaproveitadas nas diferentes atividades. Entende-se que se a atividade for fácil, as duas primeiras perguntas devem ter a maior parte das respostas como `concordo` e `concordo totalmente`, enquanto as duas últimas devem ter a maior parte em `discordo` e `discordo totalmente`.

Na Figura 6.9 vemos os gráficos das respostas, e no geral a partir dele podemos concluir que a atividade foi fácil. Apenas como ponto de atenção, na terceira pergunta 23% das pessoas indicaram terem demorado um pouco na tela inicial até encontrar onde deveriam ir. Isso pode ser um indicativo de problema na usabilidade, mas ainda é cedo pra afirmar com certeza.

Figura 6.7: Primeiro componente usado na atividade 1

Qual componente você tentou usar primeiro para realizar a atividade 1?

Opções	Total	%
Barra de busca	25	17%
Menu de tags	27	18%
Final da página (desci até exibir o post)	94	64%
	146	**100%**

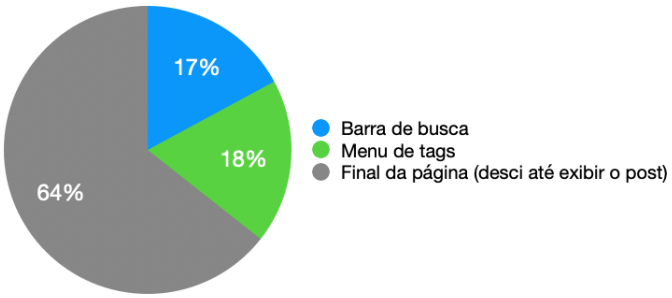
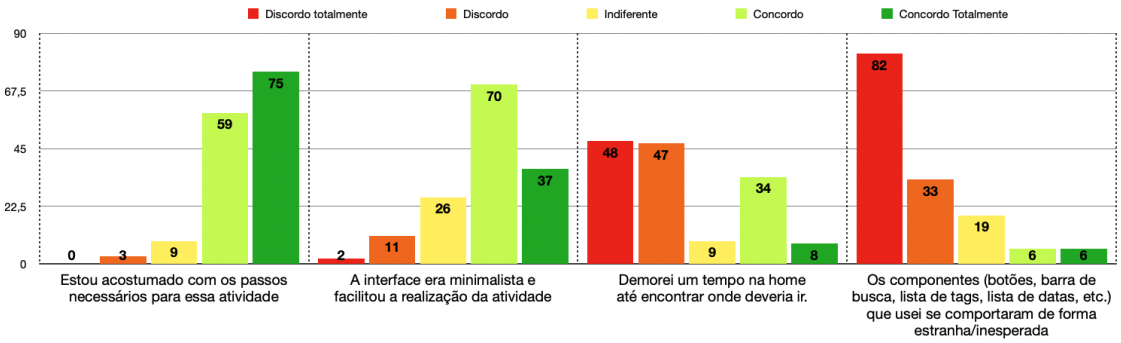


Figura 6.8: Afirmações com escala de Likert para a atividade 1

O quanto você concorda com as afirmações abaixo sobre a atividade 1?

Opções	Discordo totalmente	Discordo	Indiferente	Concordo	Concordo Totalmente
Estou acostumado com os passos necessários para essa atividade	0	3	9	59	75
	0%	2%	6%	40%	51%
A interface era minimalista e facilitou a realização da atividade	2	11	26	70	37
	1%	8%	18%	48%	25%
Demorei um tempo na home até encontrar onde deveria ir.	48	47	9	34	8
	33%	32%	6%	23%	5%
Os componentes (botões, barra de busca, lista de tags, lista de datas, etc.) que usei se comportaram de forma estranha/inesperada	82	33	19	6	6
	56%	23%	13%	4%	4%

Figura 6.9: Gráfico das afirmações em escala de Likert para a atividade 1



A última pergunta das atividades é uma questão qualitativa, diferente das anteriores. Ela serviu como espaço para comentários sobre a atividade. Foram obtidas 27 respostas, e dentre elas vale destacar as seguintes:

- "Confesso que só vi a barra de busca na terceira vez que entrei na página. Acho que ela centralizada me faria percebê-la mais rápido";
- "Tentei a `[barra de]` busca usando o título entre aspas, como geralmente funciona, mas no caso dessa atividade não retornou nenhum post e tive que retirar as aspas para funcionar";
- "Eu comecei pesquisando `[na barra de busca]` com a frase 'an introduction to kibana', ou seja, algumas palavras chaves, mas não a frase completa, nem na ordem correta; e a interface retornou 'No post found"';
- "A barra não funcionou quando pesquisei apenas uma parte do titulo";
- "No buscador o que você procura deve estar igual ao nome no post por exemplo, quando fui procurar a pagina indicada, coloquei tudo em minúsculo e deu que o arquivo não existia".

Estas respostas parecem estar alinhadas com os 23% que demoraram na *home* até encontrar onde deveriam ir. A conclusão é que identificamos alguns problemas de usabilidade na barra de busca, que deixou alguns usuários frustrados por não obterem o resultado esperado.

6.3 Atividade 2

Na atividade 2, a partir da página inicial as pessoas deveriam exibir apenas as publicações com a tag `#ios`. O comportamento esperado aqui seria de usarem o menu lateral de tags ou a barra de busca.

A Figura 6.10 mostra que a grande maioria utilizou o menu de tags, enquanto cerca de um quarto utilizou a barra de busca. Tivemos 9 pessoas que indicaram terem descido até exibir as publicações, o que indica que a pessoa ou não entendeu a pergunta ou não encontrou os componentes que levariam à resposta esperada.

A Figura 6.11 apresenta as afirmações e as respostas obtidas, enquanto a Figura 6.12 apresenta esses dados em gráfico. A distribuição de respostas está de acordo com o esperado, indicando que no geral parece não ter ocorrido alguma grande dificuldade nessa atividade.

Figura 6.10: Primeiro componente usado na atividade 2

Qual componente você tentou usar primeiro para realizar a atividade 2?

Opções	Total	%
Barra de busca	40	27%
Menu de tags	97	66%
Final da página (desci até exibir o post)	9	6%
	146	**100%**

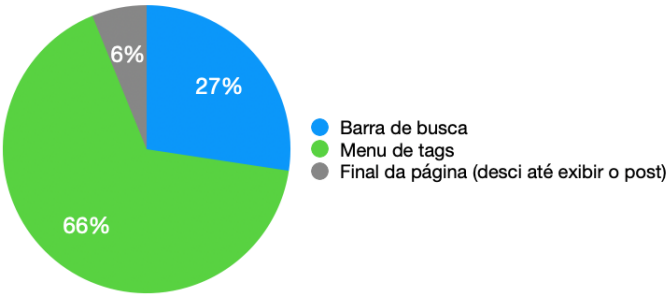
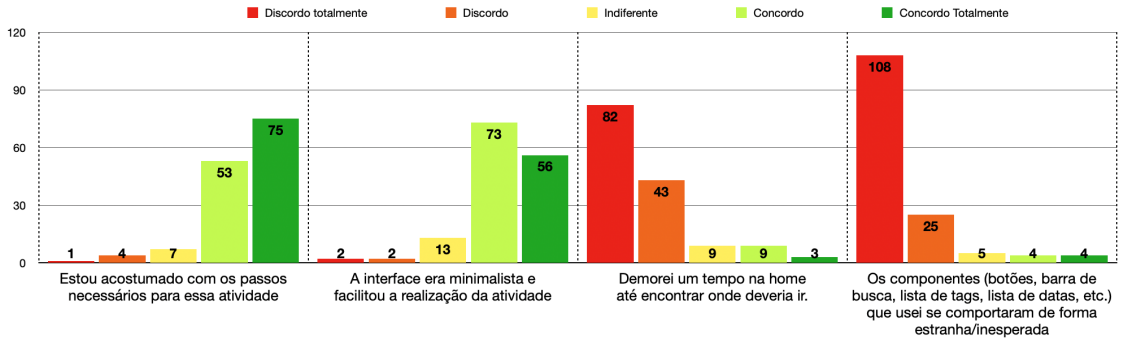


Figura 6.11: Afirmações com escala de Likert para a atividade 2

O quanto você concorda com as afirmações abaixo sobre a atividade 2?

Opções	Discordo totalmente	Discordo	Indiferente	Concordo	Concordo Totalmente
Estou acostumado com os passos necessários para essa atividade	1 1%	4 3%	7 5%	53 38%	75 54%
A interface era minimalista e facilitou a realização da atividade	2 1%	2 1%	13 9%	73 50%	56 38%
Demorei um tempo na home até encontrar onde deveria ir.	82 56%	43 29%	9 6%	9 6%	3 2%
Os componentes (botões, barra de busca, lista de tags, lista de datas, etc.) que usei se comportaram de forma estranha/inesperada	108 74%	25 17%	5 3%	4 3%	4 3%

Figura 6.12: Gráfico das afirmações em escala de Likert para a atividade 2



Sobre os comentários da atividade 2, tivemos 16 respostas, e a lista abaixo mostra algumas:

- “Quando testei com um erro de digitação ele não buscou por conteúdos similares/parecidos com a digitação. Talvez seja uma melhora futura”;
- “Uma sugestão é talvez adicionar a funcionalidade de autocomplete nessa busca. Em casos como ao digitar # (hashtag) pode ser interessante o resultado com autocomplete”;
- “A página de destino tem uma largura desconfortável de ler (estou usando computador), muito comprida. Na home, cada card de post tem uma largura um pouco extensa, dificultando a ação de scanear informações [...]”;
- “Dessa vez, inseri a frase com uma palavra incorreta ‘using criptokit’ ao invés de ‘cryptokit’. Minha expectativa era de que a interface corrigisse palavra ou desse uma sugestão de post baseado na minha entrada, visto que são palavras bem similares”.

Essas respostas parecem reforçar o problema de usabilidade na barra de busca.

6.4 Atividade 3

Na atividade 3, a partir da home, a pessoa deveria exibir apenas as postagens do ano de 2021. A hipótese era de que elas usariam o menu de datas, clicando em `2021`. A Figura 6.13 mostra que a grande maioria tentou usar o menu de datas. Porém, há 10 usuários que indicaram não terem conseguido concluir a atividade. Esse é um número significativo para uma atividade que deveria ser simples.

As Figuras 6.14 e 6.15 mostram o resultado das afirmações. Assim como na atividade 1, percebemos que há um pico de pessoas indicando que demoraram na página inicial até encontrar onde deveriam ir. Ao mesmo tempo, cerca de 10% das pessoas indicaram não estar acostumada com os passos necessários e que a interface não facilitou a realização da atividade. Por fim, vale ressaltar que cerca de 12% dos participantes tentaram utilizar a barra de busca. Porém a barra não implementa busca por data, e os usuários tentarem usar ela é um indício de que essa é uma funcionalidade necessária.

Sobre os comentários, houveram 27 respostas, e as destacadas são:

- “A facilidade que eu tive nas questões anteriores eu não tive nessa, mas apenas

Figura 6.13: Primeiro componente usado na atividade 3

Qual componente você tentou usar primeiro para realizar a atividade 3?

Opções	Total	%
Menu de datas	111	76%
Barra de busca	18	12%
Não conseguiu	10	7%
Menu de tags	7	5%
	146	**100%**

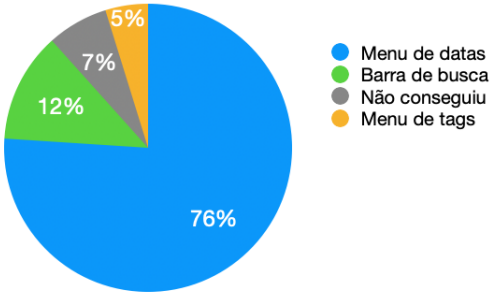
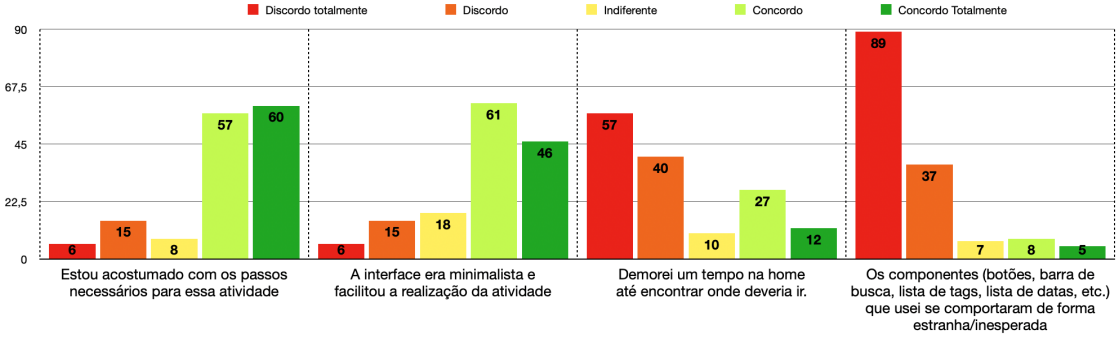


Figura 6.14: Afirmações com escala de Likert para a atividade 3

O quanto você concorda com as afirmações abaixo sobre a atividade 3?

Opções	Discordo totalmente	Discordo	Indiferente	Concordo	Concordo Totalmente
Estou acostumado com os passos necessários para essa atividade	6 4%	15 10%	8 5%	57 39%	60 41%
A interface era minimalista e facilitou a realização da atividade	6 4%	15 10%	18 12%	61 42%	46 32%
Demorei um tempo na home até encontrar onde deveria ir.	57 39%	40 27%	10 7%	27 18%	12 8%
Os componentes (botões, barra de busca, lista de tags, lista de datas, etc.) que usei se comportaram de forma estranha/inesperada	89 61%	37 25%	7 5%	8 5%	5 3%

Figura 6.15: Gráfico das afirmações em escala de Likert para a atividade 3



porque o menu de datas ficava um pouco abaixo na tela (tive que rolar pra baixo)";

- "Tentei usar a barra de busca primeiro porque na página que eu estava eu não tinha acesso ao menu de datas, depois que voltei pra home que entendi onde deveria clicar. Talvez se o menu de datas estivesse disponível em todas as páginas isso facilitasse minha interação";
- "Demorei um pouco até entender que o ano também era clicável";
- "Primeiramente procurei nas tags o ano, em seguida algum tipo de botão para ordenação dos posts. Mas logo em seguida dei um scroll e vi o widget de data";
- "Como esse filtro era muito específico realmente é normal que a pessoa precise pensar em 'como chego nisso'. Mas realmente me lembro muito pouco te já ter filtrado conteúdo por data, fora quando uso limitação de datas (limitar a pesquisa a no máximo 3 meses atrás, por exemplo)".

Com essas respostas, podemos ver que a localização do menu de datas, na lateral da tela inicial, não é tão visível quando deveria. Além disso, os usuários indicaram problemas para identificar que o ano era clicável. Dessa forma, podemos perceber um problema na usabilidade desse componente.

6.5 Atividade 4

Na atividade quatro, a partir da página inicial, as pessoas deveriam encontrar a área de comentários de qualquer publicação. As afirmações usando a escala de Likert e a distribuição de respostas podem ser vistas na tabela da Figura 6.16 e nos gráficos da Figura 6.17. No geral, essa distribuição indica que a atividade foi fácil.

Como comentários, foram deixadas 9 respostas, destacando-se as seguintes:

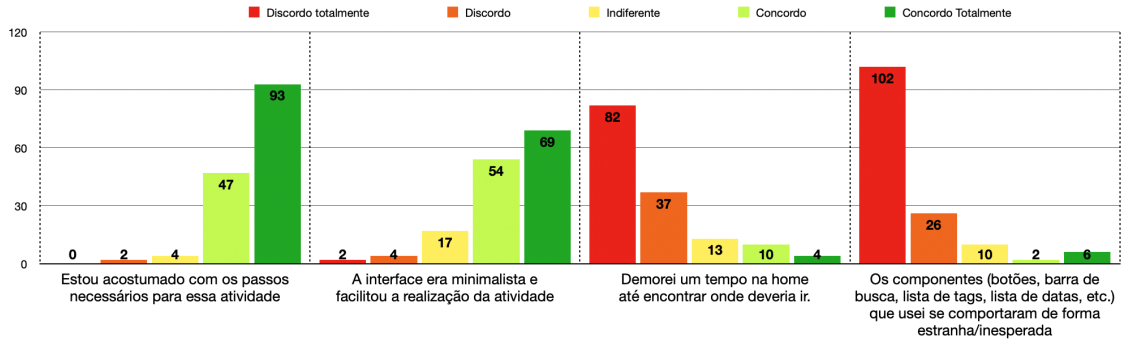
- "Ao tentar comentar, o comentário aparece mas também mostra um erro. poderia ter algo para validar que a pessoa não coloque só uma letra de comentário";
- "a página começou no fim e depois rolou para cima! Acho que deve ser um bug";
- "O título de cada post que levaria a uma postagem específica poderia ter um efeito de hover para chamar mais atenção no clique, não deixar apenas para a mudança de ponteiro do mouse indicar que era um link".
- "A visualização da postagem na home poderia ter um ícone de comentários (algo como um balão de fala, com o número de comentários ao lado) que, no clique, me

Figura 6.16: Afirmações com escala de Likert para a atividade 4

O quanto você concorda com as afirmações abaixo sobre a atividade 2?

Opções	Discordo totalmente	Discordo	Indiferente	Concordo	Concordo Totalmente
Estou acostumado com os passos necessários para essa atividade	0 0%	2 1%	4 3%	47 32%	93 64%
A interface era minimalista e facilitou a realização da atividade	2 1%	4 3%	17 12%	54 37%	69 47%
Demorei um tempo na home até encontrar onde deveria ir.	82 56%	37 25%	13 9%	10 7%	4 3%
Os componentes (botões, barra de busca, lista de tags, lista de datas, etc.) que usei se comportaram de forma estranha/inesperada	102 70%	26 18%	10 7%	2 1%	6 4%

Figura 6.17: Gráfico das afirmações em escala de Likert para a atividade 4



levaria direto para a seção de comentários no post. Assim eu não precisaria clicar em 'Read more' e rolar até encontrar a seção".

- "Um ponto importante para a usabilidade é que os campos sejam identificados de maneira fácil. No caso, o rotulo do campo foi utilizado como placeholder, dessa forma se caso eu tenha preenchido alguma coisa, mas esqueça o que é esperado por aquele campo, eu preciso remover o conteúdo para ver novamente qual é o rótulo daquele campo. Claro que no seu caso, é um formulário muito curto, o que reduz esse problema, mas mesmo assim não costuma ser uma boa prática ocultar o nome do campo quando o campo possui valor".

As duas primeiras respostas dizem respeito a possíveis problemas no código, enquanto as duas últimas se enquadram como sugestões de melhoria para a experiência do usuário.

6.6 Percepções Gerais

Na última seção do formulário foram feitas algumas afirmações sobre a experiência geral do blog. Tais afirmações e a distribuição de suas respostas são vistas nas figuras 6.18, 6.19 e 6.20.

Resumindo, os resultados obtidos nessa seção indicam que:

- 86% acreditam que a lista de tags é útil;
- 73% recomendariam o blog;
- 93% acham o blog é fácil de navegar;
- 87% pensam que o blog possui uma interface amigável;
- 63% indicam que a barra de busca se comportou como esperavam, enquanto 12% indicam o contrário;
- 59% acreditam que os botões para compartilhamento nas redes sociais são úteis, enquanto 15% acreditam que não são;
- 57% acham a lista de datas necessária, enquanto 19% acham ela desnecessária;
- 81% indicam estar claro o objetivo do blog desde a primeira página, enquanto 11% indicam o contrário.

Esta seção também teve um espaço final para comentários, mas de um ponto geral do blog. Houveram 20 respostas, e destas destacam-se abaixo:

Figura 6.18: Afirmações com escala de Likert para as percepções gerais.

O quanto você concorda com as afirmações abaixo sobre a atividade 2?

Opções	Discordo totalmente	Discordo	Indiferente	Concordo	Concordo Totalmente
A lista de tags é útil	0 0%	4 3%	17 12%	65 45%	60 41%
Eu recomendaria este blog para conhecidos que possam se interessar pelo tema	2 1%	7 5%	30 21%	76 52%	31 21%
O blog é fácil de navegar	1 1%	4 3%	5 3%	81 55%	55 38%
O blog possui uma interface amigável	1 1%	5 3%	13 9%	80 55%	47 32%
A barra de busca se comporta como eu espero	3 2%	15 10%	36 25%	53 36%	39 27%
Os botões para compartilhar em redes sociais são importantes	7 5%	14 10%	39 27%	58 40%	28 19%
A lista de datas é desnecessária	28 19%	55 38%	35 24%	19 13%	9 6%
Fica aparente desde a primeira página qual é o objetivo do site	1 1%	15 10%	12 8%	80 55%	38 26%

Figura 6.19: Gráfico das quatro primeiras afirmações das percepções gerais.

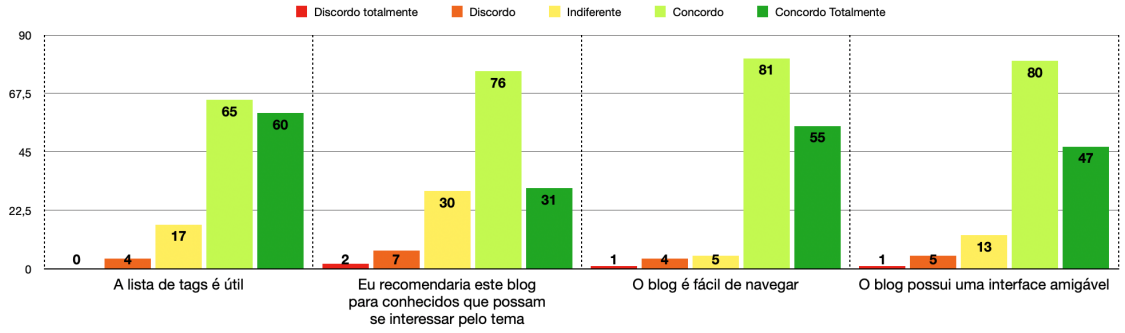
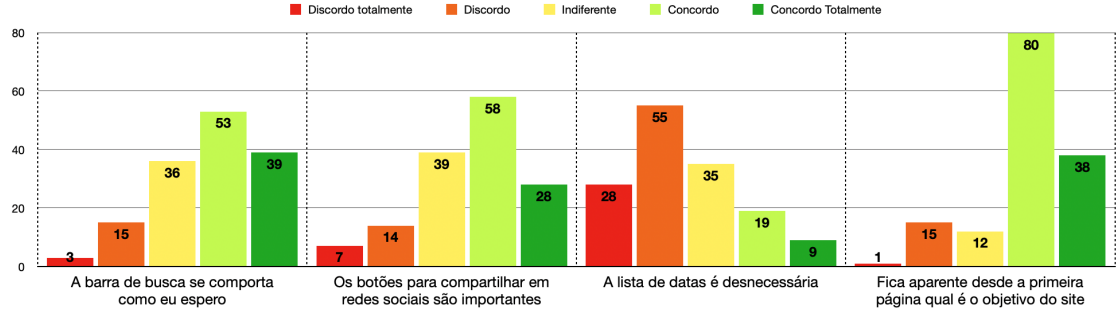


Figura 6.20: Gráfico das quatro últimas afirmações das percepções gerais.



- “Pela url eu achei que pudesse ser um currículo online. Mas dá pra saber que é um blog nos primeiros segundos que você usa a página”;
- “O layout e usabilidade são muito bem feitos e simples”;
- “Achei bem legal o blog, mas sinto que na home seria bem legal já ter um pouco mais de ‘About Me’ já que é um blog pessoal”;
- “Olhar para acessibilidade digital principalmente. Existem pontos importantes que podem impactar diferentes usuárias”;
- “Considerar a viabilidade de implementar as grafias erradas mais comuns das palavras na barra de busca”;
- “Só sobre a barra de busca, acho que um autocomplete ali enquanto preenchemos o input seria bacana, porque teríamos meio que um retorno enquanto estamos ali digitando. Mas o site em si, está bem maneiro”.

Percebemos assim que o blog parece ter uma navegação fácil, mas que alguns componentes podem ser melhorados, principalmente a barra de busca.

6.7 Análise das Respostas

A partir do formulário foram obtidos muitos feedbacks. Alguns deles foram sugestões de melhoria, outros reportaram *bugs* encontrados, enquanto a maioria indicou as dificuldades de usabilidade que encontrou. Após uma análise das respostas, três pontos se destacaram como prioritários:

- A barra de busca precisa ser mais inteligente, reduzindo os falsos negativos (quando usuário busca algo esperando encontrar uma publicação, mas ela não é retornada).
- O menu de datas precisa ser revisto. Talvez ele possa ser substituído por um filtro na barra de busca.
- O blog não está acessível para pessoas cegas, com problema de mobilidade ou que por qualquer razão navegam usando apenas o teclado. Muitos componentes precisam ser alterados para melhorar esse ponto, porém é algo importante de ser feito.

7 CONCLUSÕES

As seções abaixo discorrem sobre os resultados obtidos, as limitações da implementação e trabalhos futuros.

7.1 Resultados Obtidos

O objetivo deste trabalho era desenvolver um sistema web estilo blog usando tecnologias modernas e lidando com desafios que vão desde a configuração da infraestrutura até o projeto da interface gráfica. A intenção ao lidar com desafios diversos era aplicar conhecimentos diversos obtidos durante o curso de ciência da computação. Ao mesmo tempo, o uso de tecnologias modernas me permitiu ter contato com conhecimentos valiosos para uma carreira como engenheiro de software.

Sobre o produto final desenvolvido, foi obtido um sistema completamente apto para publicação de conteúdos relacionados à ciência da computação. Esse sistema será usado para compartilhamento de conhecimento e, idealmente, para ajudar outras pessoas em suas jornadas de aprendizado sobre ciência da computação, estimulando-as com os desafios incríveis que uma carreira nessa área oferece.

7.2 Limitações

Sobre as limitações deste trabalho, vale ressaltar o resultado obtido na avaliação (Capítulo 6). Nela percebemos problemas de usabilidade, principalmente relativos à barra de busca, ao menu de datas e à falta de acessibilidade. Destes, o mais trabalhoso pra resolver será a falta de acessibilidade, pois muitos componentes precisariam ser ajustados. Apesar disso é um trabalho importante de ser feito, e deve ser priorizado.

Outros pontos limitantes dizem respeito ao conteúdo dos menus laterais. Atualmente, o menu de tags exibe todas as tags existentes, e o menu de datas todas as datas. Essa abordagem não escala tão bem à medida em que novas tags e datas forem criadas. Uma alternativa seria exibir as tags mais usadas, e dividir as datas de outra forma, talvez numa divisão contextual como por exemplo: “último mês”, “último semestre”, “último ano” e “anterior”.

7.3 Trabalho Futuro

Como trabalho futuro, além de efetivamente utilizar o sistema desenvolvido, existem algumas melhorias e funcionalidades a serem implementadas:

Melhorias de Acessibilidade

O código que constrói a interface web precisa ser revisto, priorizando componentes semanticamente corretos. Dessa forma, o navegador facilitará a interação de usuários PcD, principalmente os que possuem alguma deficiência visual ou motora.

Melhorias na Barra de Busca

Como visto no Capítulo 6, a barra de busca foi o componente que teve mais problemas de usabilidade. Por isso, como trabalho futuro, ela será melhorada para reduzir os casos de falso negativo. Essa melhoria será feita com as seguintes mudanças:

- Permitir busca por qualquer sequência de palavras que estejam no título ou conteúdo de uma publicação;
- Entender erros de digitação, possivelmente usando algum algoritmo que compare a similaridade entre palavras;
- Permitir busca por data na barra.

Melhorias no Login

Outro ponto importante de ser melhorado é o login. Atualmente não há limite para tentativas de login, e essa é uma funcionalidade de segurança importante. Por isso como trabalho futuro o reCAPTCHA do Google será adicionado também no formulário de login.

Invalidar o Token no Logout

Atualmente quando o administrador faz *logout*, o cookie do token é apagado, porém ele continua válido no servidor. Como melhoria futura será implementado um *endpoint* que irá realizar essa invalidação do lado do servidor.

Paginação no Dashboard

Atualmente na página de *dashboard* (Seção 5.15), as publicações não estão paginadas. Como isso não escala bem, a mesma paginação usada na *home* do blog será aplicada também no *dashboard*.

Lixeira de Publicações

Outro trabalho futuro será uma melhoria no sistema de deleção de publicações. Atualmente quando elas são deletadas, elas são instantaneamente apagadas do banco de dados. Uma funcionalidade interessante é ter nesse banco uma coleção denominada lixeira, onde ficam armazenadas temporariamente as publicações deletadas. Após um período de tempo elas são efetivamente apagadas. Isso evita que seja perdida alguma publicação que foi deletada por engano.

Draft de Publicações

No blog, a tela de criação de publicações não foi implementada como um ambiente para escrita, mas apenas sim para colar o conteúdo vindo de algum editor de texto que aceite markdown (Seção 2.24). Como trabalho futuro essa tela de criação de publicações será melhorada, para que assim seja possível criar textos e mantê-los em um estado ainda não publicado lá (chamado estado de *draft*). Isso permitirá começar a usar o sistema implementado também para escrita.

Armazenamento de Imagens das Publicações

Atualmente, as imagens visíveis nas publicações são apenas URLs que apontam para sistemas de armazenamento terceiros (e.g., Google Drive). Como melhoria futura será usado o S3 (Seção 2.4) para manter essas imagens. Assim o sistema não dependerá mais de armazenamento em serviço externo.

REFERÊNCIAS

AMAZON. **Amazon Containers customers**. 2021. Acesso em: 13 de out. de 2021. Available from Internet: <https://aws.amazon.com/containers/customers/>.

AMAZON. **Amazon DynamoDB customers**. 2021. Acesso em: 13 de out. de 2021. Available from Internet: <https://aws.amazon.com/dynamodb/customers/>.

AMAZON. **Amazon EC2 customers**. 2021. Acesso em: 13 de out. de 2021. Available from Internet: <https://aws.amazon.com/ec2/customers/>.

AMAZON. **Amazon S3 customers**. 2021. Acesso em: 13 de out. de 2021. Available from Internet: <https://aws.amazon.com/s3/customers/>.

AMAZON. **AWS. Instâncias T2 do Amazon EC2**. 2021. Acesso em: 31 de ago. de 2021. Available from Internet: <https://aws.amazon.com/pt/ec2/instance-types/t2/>.

AMAZON. **AWS Website**. 2021. <https://aws.amazon.com/>. Acessado dia 25 de out. de 2021.

ARAVINDNC. **Mongoose Paginate v2**. 2021. <https://github.com/aravindnc/mongoose-paginate-v2>. Acessado dia 25 de out. de 2021.

AUTH0. **JSON Web Tokens**. Acesso em: 19 de out. de 2021. Available from Internet: <https://jwt.io/>.

AUTH0. **Express JWT**. 2021. <https://github.com/auth0/express-jwt>. Acessado dia 25 de out. de 2021.

AUTH0. **JWT Website**. 2021. <https://jwt.io/>. Acessado dia 25 de out. de 2021.

AWS. **Amazon EC2**. 2021. <https://aws.amazon.com/ec2>. Acessado dia 25 de out. de 2021.

AWS. **Amazon S3**. 2021. <https://aws.amazon.com/s3/>. Acessado dia 25 de out. de 2021.

AWS. **AWS SDK Github**. 2021. <https://github.com/aws/aws-sdk>. Acessado dia 25 de out. de 2021.

AWS. **What is DevOps**. 2021. <https://aws.amazon.com/devops/what-is-devops>. Acessado dia 25 de out. de 2021.

AXIOS. **Getting Started with Axios**. 2021. <https://axios-http.com/docs/intro>. Acessado dia 25 de out. de 2021.

AXIOS. **Repositório do Axios no Github**. 2021. <https://github.com/axios/axios>. Acessado dia 25 de out. de 2021.

BLUEHOST. **What is Rest**. 2021. <https://restfulapi.net/>. Acessado dia 25 de out. de 2021.

BOOTSTRAP. **Bootstrap Website**. 2021. <https://getbootstrap.com/>. Acessado dia 25 de out. de 2021.

BOOTSTRAP, R. **React Bootstrap Github**. 2021. <https://github.com/react-bootstrap/react-bootstrap>. Acessado dia 25 de out. de 2021.

CAROLI, P. **Lean Inception: How to Align People and Build the Right Product**. Editora Caroli, 2019. ISBN 9788594377135. Available from Internet: <https://books.google.com.br/books?id=gL6BywEACAAJ>.

COMMUNITY, E. **Express Documentation**. 2021. <https://expressjs.com/>. Acessado dia 25 de out. de 2021.

COMMUNITY, S. C. **Styled Components Website**. 2021. <https://styled-components.com/>. Acessado dia 25 de out. de 2021.

COMUNITY, S. **Sass Website**. 2021. <https://sass-lang.com/>. Acessado dia 25 de out. de 2021.

CONSERVANCY, S. F. **Git Website**. 2021. <https://git-scm.com/>. Acessado dia 25 de out. de 2021.

DOZOISCH. **React Google Recaptcha Github**. 2021. <https://github.com/dozoisch/react-google-recaptcha>. Acessado dia 25 de out. de 2021.

EXPRESSJS. **Morgan Github**. 2021. <https://github.com/expressjs/morgan>. Acessado dia 25 de out. de 2021.

FACEBOOK. **Open Graph Protocol**. 2021. <https://ogp.me/>. Acessado dia 25 de out. de 2021.

FIELDING, R. T. **Representational State Transfer (REST)**. 2021. <https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm>. Acessado dia 25 de out. de 2021.

FORMIDABLE node. **Formidable Github**. 2021. <https://github.com/node-formidable/formidable>. Acessado dia 25 de out. de 2021.

FOUNDATION, E. F. **Certbot**. 2021. <https://certbot.eff.org/>. Acessado dia 25 de out. de 2021.

FOUNDATION, O. **Node JS Dev Website**. 2021. <https://nodejs.dev/>. Acessado dia 25 de out. de 2021.

FOUNDATION, O. **Node JS Website**. 2021. <https://nodejs.org/en/>. Acessado dia 25 de out. de 2021.

GA-MO. **React Confirm Alert Github**. 2021. <https://github.com/GA-MO/react-confirm-alert>. Acessado dia 25 de out. de 2021.

GITHUB. **Github Website**. 2021. <https://github.com/>. Acessado dia 25 de out. de 2021.

GRUBER, J. **Markdown**. 2004. <https://daringfireball.net/projects/markdown/>. Acessado dia 25 de out. de 2021.

HELLER, E. **A Psicologia das cores: como as cores afetam a emoção e a razão**. Editorial Gustavo Gili, SL, 2014. ISBN 9788565985079. Available from Internet: <https://books.google.com.br/books?id=l6rYNAEACAAJ>.

HOLLAND, A. **How Estimated Reading Times Increase Content Engagement**. 2014. <https://martech.org/estimated-reading-times-increase-engagement/>. Acesso em: 20 de out. de 2021.

IETF. **RFC-6234**. 2021. <https://datatracker.ietf.org/doc/html/rfc6234>. Acessado dia 25 de out. de 2021.

(ISRG), I. S. R. G. **Let's Encrypt Website**. 2021. <https://letsencrypt.org/>. Acessado dia 25 de out. de 2021.

JONES, M.; BRADLEY, J.; SAKIMURA, N. **JSON Web Token (JWT)**. [S.l.], 2015. <http://www.rfc-editor.org/rfc/rfc7519.txt>. Available from Internet: <http://www.rfc-editor.org/rfc/rfc7519.txt>.

JS-COOKIE. **JS-Cookie Github**. 2021. <https://github.com/js-cookie/js-cookie>. Acessado dia 25 de out. de 2021.

JS, E. **Express Cors Github**. 2021. <https://github.com/expressjs/cors>. Acessado dia 25 de out. de 2021.

JS, M. **Mongoose Website**. 2021. <https://mongoosejs.com/>. Acessado dia 25 de out. de 2021.

JS, N. **Next JS Website**. 2021. <https://nextjs.org/>. Acessado dia 25 de out. de 2021.

JS, R. **React JS Website**. 2021. <https://reactjs.org/>. Acessado dia 25 de out. de 2021.

KIM, L. **The Psychology of Logo Color in How Consumers View Your Brand**. 2016. Available from Internet: <https://medium.com/marketing-and-entrepreneurship/the-psychology-of-logo-color-in-how-consumers-view-your-brand-d3afe84f2bdb>.

LEARN Faster with The Feynman Technique. 2011. Acesso em: 24 de out. de 2021. Available from Internet: <https://youtu.be/FrNqSLPaZLc>.

MARKED. **Marked Documentation**. 2021. <https://marked.js.org/>. Acessado dia 25 de out. de 2021.

MARKED. **Marked Github**. 2021. <https://github.com/markedjs/marked>. Acessado dia 25 de out. de 2021.

MEDIUM. 2012. Acessado dia 20 de out. de 2021. Available from Internet: <https://medium.com/>.

MONGODB, I. **MongoDB Atlas Website**. 2021. <https://www.mongodb.com/atlas/database>. Acessado dia 25 de out. de 2021.

MONGODB, I. **MongoDB Website**. 2021. <https://www.mongodb.com/>. Acessado dia 25 de out. de 2021.

MONGODB, I. **What is NoSQL?** 2021. <https://www.mongodb.com/nosql-explained>. Acessado dia 25 de out. de 2021.

MOTDOTLA. **Dotenv Github**. 2021. <https://github.com/motdotla/dotenv>. Acessado dia 25 de out. de 2021.

MOZILLA. **Express/Node Introduction**. 2021. <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction>. Acessado dia 25 de out. de 2021.

MOZILLA. **HTTP**. 2021. <https://developer.mozilla.org/en-US/docs/Web/HTTP>. Acessado dia 25 de out. de 2021.

MOZILLA. **Identifying resources on the Web**. 2021. <https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Identifying_resources_on_the_Web>. Acessado dia 25 de out. de 2021.

NAH, F. F.-H. A study on tolerable waiting time: how long are web users willing to wait? **Behaviour & Information Technology**, Taylor & Francis, v. 23, n. 3, p. 153–163, 2004. Available from Internet: <https://doi.org/10.1080/01449290410001669914>.

NETWORKS, F. **NGINX Website**. 2021. <https://www.nginx.com/>. Acessado dia 25 de out. de 2021.

NPROGRESS. **NProgress Github**. 2021. <https://github.com/rstacruz/nprogress>. Acessado dia 25 de out. de 2021.

NYGARDK. **React Share Github**. 2021. <https://github.com/nygardk/react-share>. Acessado dia 25 de out. de 2021.

OAKLEY, B. **A Mind For Numbers: How to Excel at Math and Science (Even If You Flunked Algebra)**. Penguin Publishing Group, 2014. ISBN 9781101621615. Available from Internet: <https://books.google.com.br/books?id=Sb1iAgAAQBAJ>.

PRISMJS. **Prism Github**. 2021. <https://github.com/PrismJS/prism>. Acessado dia 25 de out. de 2021.

PRISMJS. **Prism Website**. 2021. <https://prismjs.com/>. Acessado dia 25 de out. de 2021.

REMY. **Nodemon Github**. 2021. <https://github.com/remy/nodemon>. Acessado dia 25 de out. de 2021.

ROGERS, Y.; SHARP, H.; PREECE, J. **Design de interação: além da interação humano-computador**. Bookman, 2013. ISBN 9788582600061. Available from Internet: <https://books.google.com.br/books?id=wERUmgEACAAJ>.

SETUP Node.js + MongoDB Production Server on Ubuntu 18.04 - Ubuntu 19.04. Available from Internet: <https://jasonwatmore.com/post/2018/09/26/setup-nodejs-mongodb-production-server-on-ubuntu-1804>.

UNGER, R.; CHANDLER, C. **A Project Guide to UX Design: For User Experience Designers in the Field Or in the Making**. New Riders, 2009. (Voices that matter). ISBN 9780321607379. Available from Internet: <https://books.google.com.br/books?id=7IkqAQAAMAAJ>.

UNIVERSITY, N. D. of M. **History of Blogging**. 2018. <https://online.ndm.edu/news/communication/history-of-blogging/>. Acessado dia 25 de out. de 2021.

VALIDATOR express. **Express Validator Github**. 2021. <https://github.com/express-validator/express-validator>. Acessado dia 25 de out. de 2021.

WIGGINS, A. **The Twelve-Factor App: III Config**. 2017. <https://12factor.net/config>. Acessado dia 30 de out. de 2021.

WIKIPEDIA. **A wireframe document for a person profile view**. 2009. <https://en.wikipedia.org/wiki/Website_wireframe#/media/File:Profilewireframe.png>. Acessado dia 25 de out. de 2021.

WIKIPEDIA. **Blog**. 2021. <https://en.wikipedia.org/w/index.php?title=Blog&oldid=1051180806>. [Online; accessed 25-October-2021].

WIKIPEDIA. **Markdown**. 2021. <https://en.wikipedia.org/w/index.php?title=Markdown&oldid=1051112651>. Acessado dia 25 de out. de 2021.

WYNGAARD, C. J. V.; PRETORIUS, J. H. C.; PRETORIUS, L. Theory of the triple constraint — a conceptual review. In: **2012 IEEE International Conference on Industrial Engineering and Engineering Management**. [S.l.: s.n.], 2012. p. 1991–1997.

YOUNG, S.; CLEAR, J. **Ultralearning: Master Hard Skills, Outsmart the Competition, and Accelerate Your Career**. Harper Business, 2019. ISBN 9780062852748. Available from Internet: <https://books.google.com.br/books?id=jyV2DwAAQBAJ>.

Anexos

ANEXO A — WIREFRAMES E RESULTADOS FINAIS

Figura A.1: Wireframe e resultado final da página inicial no desktop

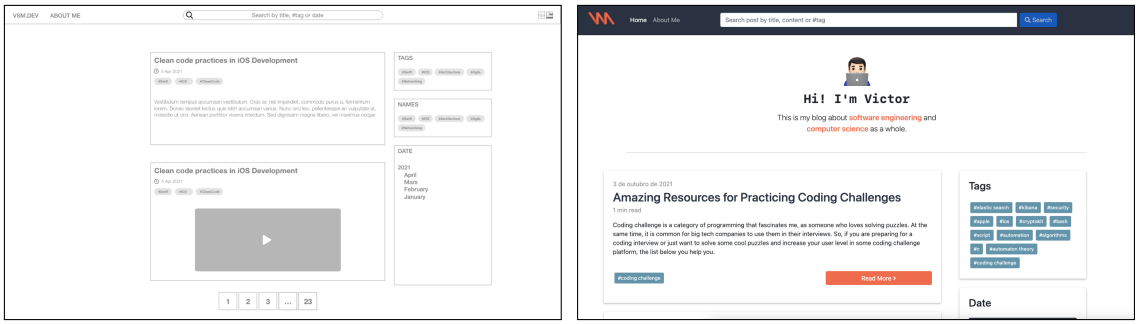


Figura A.2: Wireframe e resultado final da página inicial no mobile

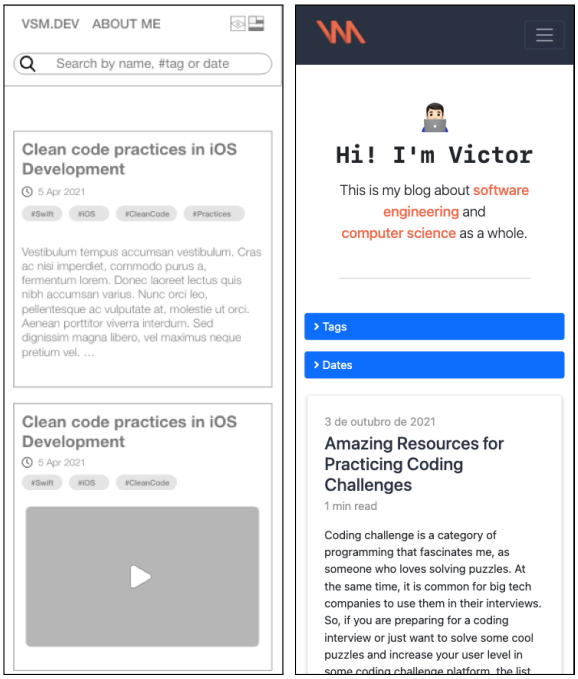


Figura A.3: Wireframe e resultado final da página de publicação no desktop

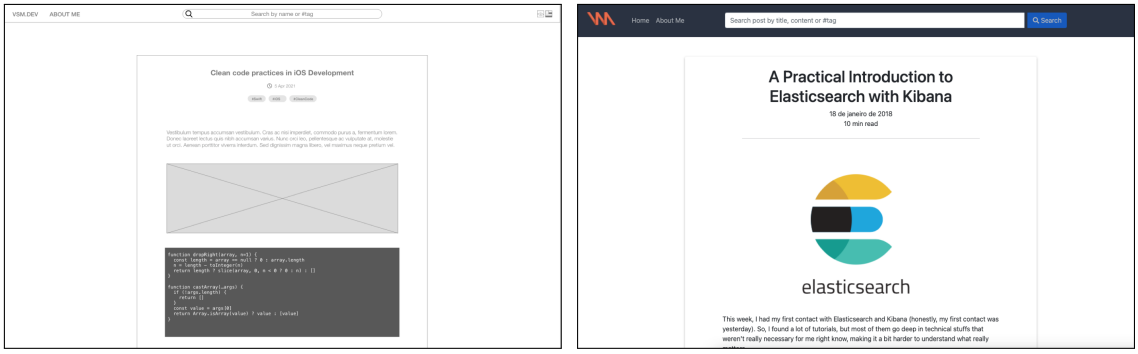


Figura A.4: Wireframe e resultado final da página de publicação no mobile

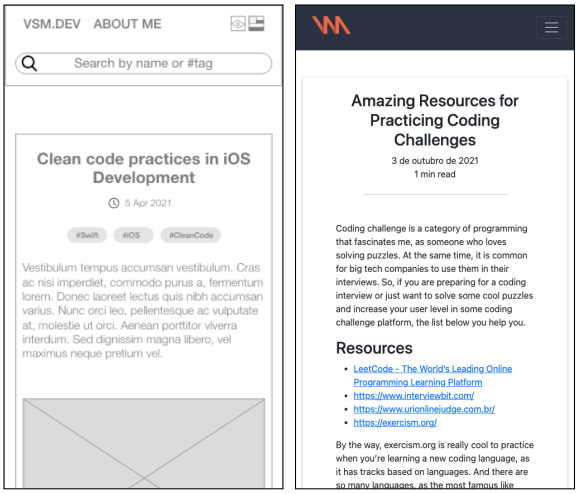


Figura A.5: Wireframe e resultado final da página "about me" no desktop

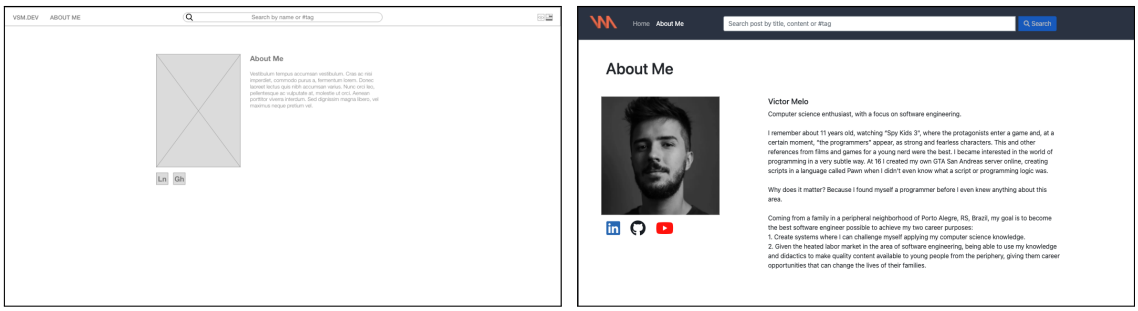


Figura A.6: Wireframe e resultado final da página “about me” no mobile

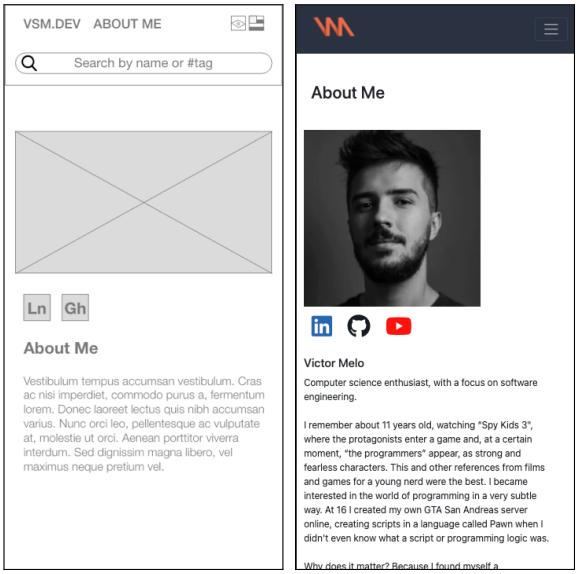


Figura A.7: Wireframe e resultado final da página de *dashboard* no desktop

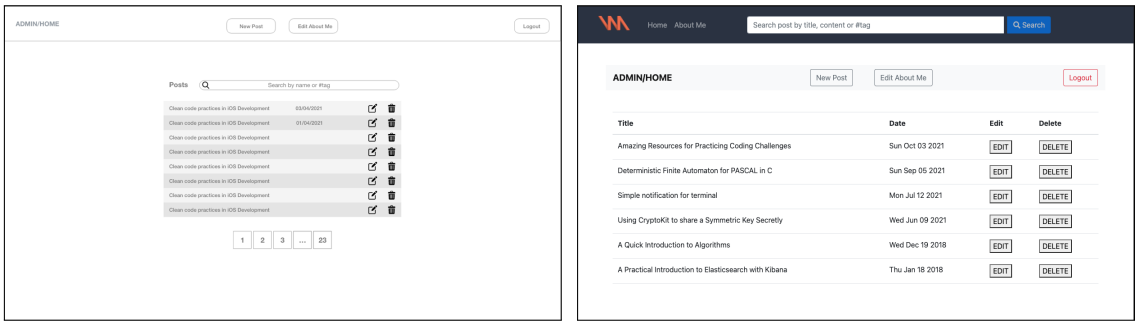


Figura A.8: Wireframe e resultado final da página de *dashboard* no mobile

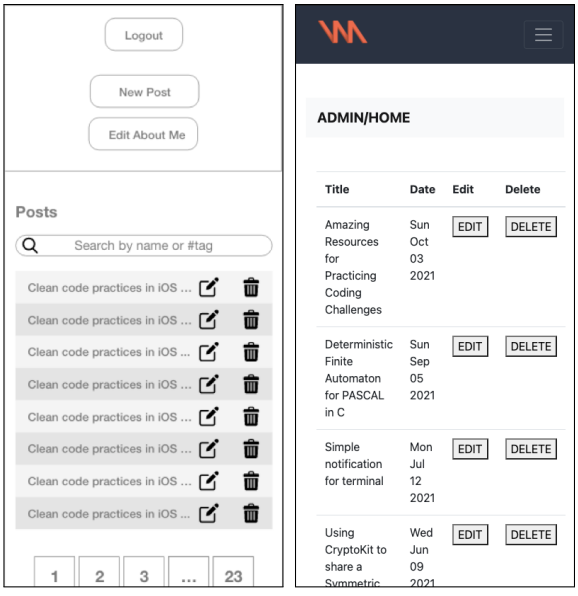


Figura A.9: Wireframe e resultado final da página de login no desktop

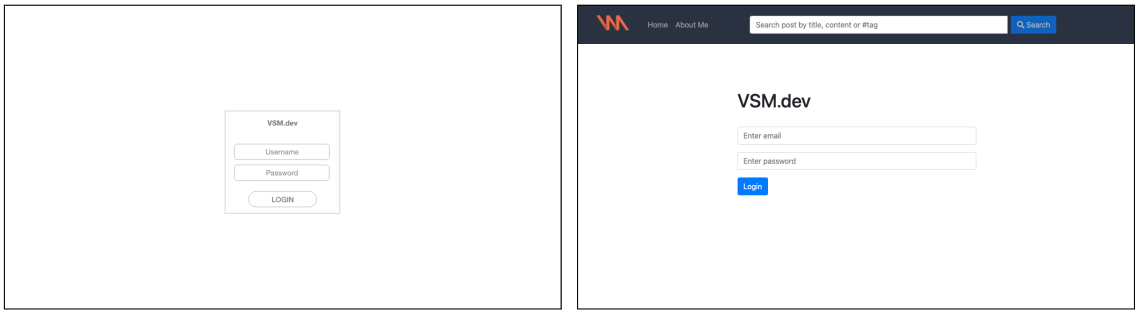


Figura A.10: Wireframe e resultado final da página de login no mobile

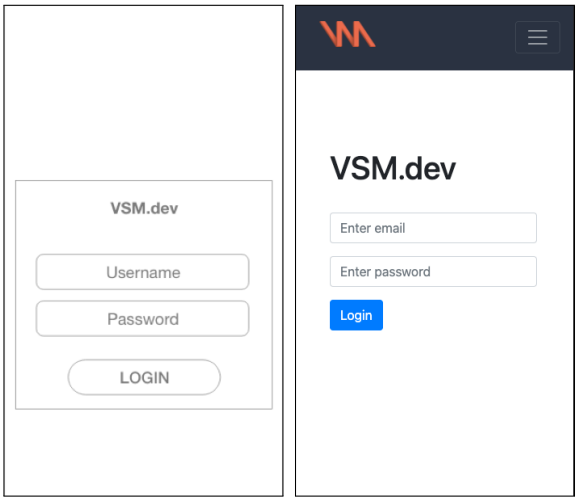
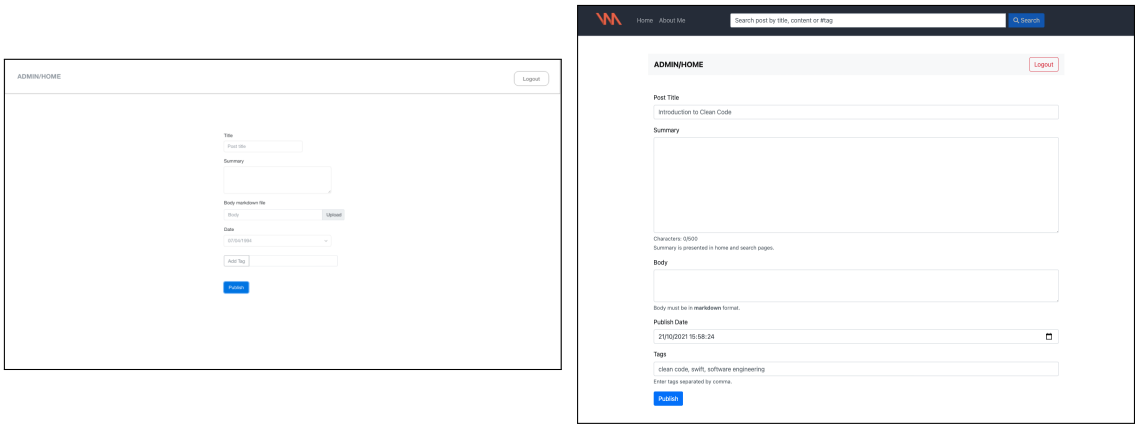


Figura A.11: Wireframe e resultado final da página de criar publicações no desktop



ANEXO B — SCRIPT DEPLOY.SH

```
# Stop and delete PM2
pm2 stop all;
pm2 delete all;

# Update backend
cd /opt/back-end;
git pull;

# Start PM2 Backend
cd /opt/back-end;
pm2 start server.js;

# Update frontend
cd /opt/front-end;
git pull;

# Build frontend
cd /opt/front-end;
sudo npm run build;

# Start PM2 Frontend
cd /opt/front-end;
pm2 start yarn --name "nextjs" --interpreter bash -- start;

# Restar nginx
sudo systemctl restart nginx;
```